TAPR

16th

ARRL and TAPR DIGITAL COMMUNICATIONS CONFERENCE

Baltimore, Maryland
October 10-12, 1997

16th ARRL and TAPR DIGITAL COMMUNICATIONS CONFERENCE

Local Co-Host: **Amateur Radio Research and Development Corporation (AMRAD)**

Conference Coordinators:

Greg Jones, WD5IVD
Paul Rinaldo, W4RI
Steve Stroh, N8GNJ
Steve Ford, WB8IMY

American Radio Realay League, Inc.
225 Main Street
Newington, CT 06111-1494 USA
tel: 860-594-0200 WWW: http://www.arrl.org/

Tucson Amateur Packet Radio
8987-309 E. Tanque Verde Rd #337
Tucson, Arizona 85749-9399 USA
tel: 940-383-0000 WWW: http://www.tapr.org

ISBN: O-87259-636-2

ARRL Order Number: 6362

First Edition

Welcome!

This is an exciting year to be a ham--particularly if you're a ham with a passion for digital communication. Only the most naive among us would deny that the future of radio is digital. As hams we're poised to take advantage of the latest ideas and technologies the digital world has to offer.

The sixteenth ARRL and TAPR Digital Communications Conference is overflowing with intriguing possibilities for Amateur Radio. The Automatic Packet Reporting System is the hottest thing in packet today, and in these proceedings you'll find new innovations to make it even more attractive. Can we reinvigorate "traditional" packet radio? Take a look at the ideas put forward by John Hansen, WAOPTV, concerning a ham version of the World Wide Web. Powerful PCs and soundcards are commonplace. Can we put them to work as digital communication devices? Several hams have done exactly that! And what of spread spectrum? As you'll read here, hams are moving from the theoretical to the practical.

These are just a few of the ideas you'll find presented on the following pages. Read and enjoy, but most of all, do. Take what you discover and apply it in the real world. That's the only way Amateur Radio can move forward as we approach the new century.

David Sumner, K1ZZ
ARRL Executive Vice President

October 1997

Table of Contents

Amateur Radio and the Linux Operating System 8/18/97

John B. Bandy, WOUT, ex-WNOTSK; Wichita, Ks U.S.A

Abstract.

This paper is about moving from MS Windows 3.1 & MS-DOS to MIT X Window & Linux 1.2.13. These operating systems run on the author's PC Intel chips supporting amateur radio applications.

Key Words.

MIT X Window, MS Windows 3.1, MS-DOS, Linux 1.2.13, PC, TAPR/AMSAT DSP-93
amateur radio packet, digital communications, AX25, TCP/IP.

Introduction.

This will be a comparsion based on experience of the two above mentioned operating systems. The author tried to be as objective as possible by pointing out the plusses and minuses of both systems. Also are lists of ham radio application software, journals, books, and Internet Sites available for Linux.

Pronouncing Linux.

The "i" in Linux is short as it is in Linus.

Comparsions.

	****** Operating System(OS) *******			
Characteristic	X Window & Linux		Windows3.1 & DOS	
plug and play	no	no	yes	yes
non-programmer friendly	yes	no	yes	no
C programmer friendly [5]	yes	yes	no	no
OS Source code available	yes	yes	no	no
runs on the PC	yes	yes	yes	yes
reliable [4]	yes	yes	somewhat	yes
network operating system [4]	yes	yes	no	no
build-in amateur radio AX25 [1][2]	yes	yes	no	no
ham applications software [6]	some	some	much	much
application freeware [6]	much	much	some	some
application shareware	some	some	much	much
application commercialware	much	much	much	much
operating system freeware.	yes	yes	no	no

hardware vendor provides drivers	no	no	yes	yes
hardware driver specs available[9].....	some	some	some	some
scripting	yes	yes	no	no
piping	yes	yes	no	no
development toolkits freeware [7].	yes	yes	no	no
runs Unix application software	.yes	yes	no	no
build-in TCP/IP [I]	yes	yes	no	no
multi-user	yes	yes	no	no
multi-tasking	yes	yes	no	no
case sensitive	yes	yes	no	no
implements a superset of the POSIX public standard [8].	..ves	yes	no	no
public design specifications..	.yes	yes	no	no

Note: The Windows NT and X Window & Linux systems probably compare closer because both are network operating systems, but the author has not experienced the Windows NT system. Linux runs on the PowerPC [3] and other platforms [9], but again the author has not experienced it.

Notes.

Notes made while moving software and data from MS Windows 3.1 & MS-DOS to MIT X Window & Linux 1.2.13 for those hams wanting to avoid the struggle. Other notes also included.

1. Stable Kernel Making

/usr/src/linuxelf- I.2.13 # make config — makes 'c' files list
make dep — establishes dependent files
make clean — removes old kernel files
make zImage — compiles 'c' code
make zdisk — creates boot disk
rdev -R /dev/fd0 1
make zlilo
/usr/lib/lilo# lilo — updates lilo

drivers in /usr/src/linuxelf- 1.2.13/drivers/

Once the 'c' files list has been established changes can be made to the 'c' code such as a driver and the kernel process started at 'make zImage'.

Soundblaster Pro 2 IRQ is 5.

2. To move something from the MS-DOS machine to the Linux machine do

in MS DOS
gzip something. c
rename something.cz somethingc.gz
rawrite somethingc.gz writes over any dos format stuff - Linux format
in LINUX
cd /usr/src/
cp /dev/fd0 something.c.gz copies entire disk
gunzip something.c.gz decompress file and skips garbage
vi something. c

3. To I/O a 3.5 diskette written on in a MS-DOS format do the following:
   a. Put diskette in drive
   b. mcopy a: * /home/jbbandy/ msdos to linux - no mounting
   c. go to linux directory and gzip -d sep_96.z or gunzip sep_96.z
   d. cannot copy a directory to a directory. Can mcd down to the file level and mcopy a: * /directory/directory .

4. To review logon messages enter “dmesg | more” less quotes.

5. Name colors denote different permission combinations:

white	data
dark blue	directory
green*	executable shell script
red	compressed
light blue@	link

6. Arguments for commands can be enclosed in apostrophes or quotes, or both.
   If an apostrophe(s) is part of a literal use quotes.
   example: sed -e "s,^${1-.},," -e '/^$/d'

7. To add a directory to the search PATH, vi /etc/profile.
   Save the old profile before modifying it.

8. chmod 777 * will change permission of all files in the directory.

9. Mouse is connected to connector marked com1/com3 on i/o board.

10. Serial port is connected to connector marked com2/com4 on i/o board.

11. When the fsck command upon startup issues the “duplicate/bad blocks” message, run “fsck -r /dev/hda1" and answer "y" to all questions at the (none): prompt (root/logout).
    Look in the lost+found file.
    Note the hda1 is mounted “read only”.

12. Run /etc/rc.d/rc.serial to get settings of internal & external serial ports.

13. Tried irq 5 on cua3, but it slowed down bps to about 300. Slow! ! Put rc.serial into rc.local.

14. lnstalled an IBM modem/sound dsp combo board, but the minicom program would not initialize it. Also the serial port line did not display when starting the computer.

15. gunzip & tar -vxf` filename compressed files suffixed with .z, .gz, or .tgz in root.
example: gunzip gw4pts_m.z
tar -vxf` gw4pts_m
gunzip sccw_tar.gz
tar -vxf` sccw.tar

When using 'gunzip', if the message . . . .more than one entry.. is received, use the 'unzip -aL' command instead of 'gunzip'.

16. To change date/time enter mmddhhmmyy. Ex. Jan 2, 1997 03:04 UTC
Seconds will show, but do NOT enter them mmddhhmmyy
0102030497

17. To remove the MS-DOS end-of-line symbol (^M) from a text file do this in root.
1. zip tncinit tncinit. txt
2. unzip -aL tncinit .zip

18. For syslog, etc. messages do these in "root":
a. dmesg | more
b, vi /usr/adm/syslog
c. vi /usr/adm/messages
d. vi /usr/adm/debug

19. 'chown -R jbandy.users *' will change owner for all sub-directories and related files. Note: '-R' stands for recursive.

20. If math routines such as sqrt, sin, tan, etc. are involved in a compile -lm must be used as a parameter in gcc. -1 means library.
example: gcc -0 color colort . c -1X 11 -lm

2 1. Cross Reference

GNU	SYSTEM V
gawk	awk
yacc	yacc
flex	lex
terminfo	terminfo
	lint
gdb	sdb
make	make
Makefiles	Makefiles
RCS	SCCS
ncurses	curses

22. "man" page names for curses/ncurses functions must be prefixed with curs . Sometimes this may not work when one "man" page covers 2 or more functions. Use the cross reference table in "man ncurses" if the "man" page appears to be missing.

23. If "call waiting" is removed "*70," number prefix must be removed or minicom gets a busy signal.

Books/Journals.

Linux Journal, SSC, Seattle, WA, phone 206-782-7733.

Eric F. Johnson & Kevin Reichard, X Window Applications Programming, MIS:Press, N.Y., N.Y., 1992.

Eric F. Johnson & Kevin Reichard, Advanced X Window Applications Programming, M&T Books, N.Y., N.Y., 1994.

Eric F. Johnson & Kevin Reichard, Professional Graphics Programming in the X Window System, MIS:Press, N.Y ., N.Y., 1993.

Rebecca Thomas & Rik Farrow, UNIX Administration Guide for System V, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

Mitchell Waite, Stephen Prata, & Donald Martin, The Waite Group's UNIX System V Primer, SAMS, Carmel, Indiana, 2nd Edition, 1992.

Stephen Prata, Donald Martin, & The Waite Group, UNIX System V Bible-Commands and Utilities, SAMS Publishing, Indianapolis, Indiana, 1987.

J. Purcell & A. Robinson, compilers, Dr. Linux - The Complete Linux reference documentation, Red Hat Software, Inc. and LSL, Westport, Ct, 4th Edition, 1996.

Ian Wade, G3NRW, NOSintro - TCP/IP over Packet Radio, Dowermain Ltd, Luton, Bedfordshire, U. K., I 992.

Kamran Husain, Timothy Parker, Ph.D; et al, Linux Unleashed, Sams Publishing, Indianapolis, IN, 2nd edition, 1996.

Note: Some Linux books are nothing more than bound doc prints from Internet Site http://sunsite.unc.edu/pub/Linux/docs and its mirrors.

Note: Most of the information in UNIX books/journals/courses/seminars also applys to Linux.

Linux Internet Address.

http://sunsite.unc.edu/pub/Linux
ftp://sunsite.unc.edu/pub/Linux
ftp://ftp.ucsd.edu/hamradio/packet/tcpip/linux
http://www.rahul.net/perens/HamRadio/LinuxAndAmateurRadio/html
http://www.arrl.org
http://www.amsat.org
ftp://ftp.amsat.org
http://www.tapr.org
ftp://ftp.tapr.org

Some Linux Ham Software.[6]

See reference for details.

Microsoft Ground Station Software - X Window xpb, xpg, xtlm.
SatTrack - X Window Satellite tracking program.
Predict - Satellite orbital prediction program.
UO1 1 - Decodes OSCAR 11 telemetry.
Dove - Decodes OSCAR 17 telemetry.
Kepgen - Keplerian elements keyboard entry program.
FT-890 - Remote Control program for Yaesu FT-890 transceiver.
F6FBB Packet BBS - X Window Bulletin Board System (BBS).
DX Cluster Watcher.
Digiinfo - Reads a database of packet radio networks information.
BayBox BBS - Packet bulletin board system.
JNET - KA9Q Network Operating System (NOS) derivative.
XNET - X Window Traffic analyzer for AX.25 networks. [1 1]
monax25 - Utilities for collecting statistics on AX.25 channel usage.
split screen - splits screen for user to usr chats.

talk-ax25 - 'talk' program to support AX.25 operation.
7Plus for Linux - encodes binary files for AX.25 messaging.
TNOS - NOS derivative.
NOARY Packet BBS for UN*X - Command set similiar to RLI bbs and extended. Ham network to INTERNET message gateway.
LBBS - BBS message gateway. SMTP/NNTP<>BBS gatewaying.
MBL/RLI message to NNTP and email converter - Converts MBL/RLI messages to either the NNTP format or RFC-822 format.
CLX Packet DX Cluster Program - Network node software that clones PacketCluster nodes.
DPTNT Terminal and DPBOX BBS package - full-featured terminal and BBS system.
IPIP Encapsulation daemon - IPIP encapsulating gateway to the Internet.
AXIP Encapsulation daemon - AXIP encapsulating gateway to the Internet.
Ping-Pong Convers Server - Roundtable chat server.
RSPF Daemon - Radio Shortest Path First (RSPF) router.
Michael Westfall's TTYLINK Daemon - Listens on the TTYLINK port and accepts connect requests.
Craig Small's TTYLINK Daemon - Answers TTYLINK requests.
Pileup - sents Morse code pileups from a sound card for copy practice.
bip - X Window sents Morse code for copy practice. Also works without X Windows and soundcard.
cw2hex - Creates an Intel hex file from keyed input for use in burning an eprom.
SoundCard CW - sents Morse code for copy practice.
GW4PTS Morse trainer - sends Morse code for copy practice from internal PC spkr.
morse (aka superiormorse) - X Window sends Morse code for copy practice.
oscope - Makes the platform function as an audio oscilloscope using the 8.8khz-44khz sound card, SVGA monitor, X Window, etc..
Software Oscilloscope - Makes the platform function as an audio oscilloscope using the sound card, SVGA monitor, etc..
Printed Circuit Board Design Tool - Layout circuit with popular component symbols using X Window.
Chipmunk Circuit Design and Simulation Tool - X Window application.
irsim - X Window MOS circuit simulator.
Spice - Analog circuit emulator. [13][14]
svgaafft - Turns the X Window platform into a audio spectrum analyzer using the sound card, monitor, etc..
Audio Spectrum Analyzer - X Window application.
ObjectProDSP - X Window Digital Signal Processing network design tool and simulator.
bpf - two pole bandpass filter calculator.
twclock - X Window displays time of day in various parts of the world.
Chirp - ncurses contest logging program.
Contest - contest logging program.
SunClock - X Window displays date, local time, UTC, and the sunny parts

of the globe.

Xearth - X Window displays the earth as view from space and the mercator projection view.

Additional Linux Ham Software.

QRZ! Ham Radio CDROM-Callsign Database - searches a call sign database.
http://www.cdrom.corn

dspload - Loads assembled dsp programs in the TAPR-AMSAT DSP-93 modem.
http://www.tapr.org

TASM - A Table Driven Cross Assembler for the Linux Environment supporting 6502,
6800/6801/68HC11/6805/TMS32010/TMS320C25/TMS7000/8048/8051/8080/8085/Z80 Microprocessor families, (alpha version as of 2/6/97 j.
Tom@main.fs1.InControl.corn
Speech Technology, 837 Front St. S., Issaquah, WA 98027

nasawash - extracts and files NASA format 2-line Kep elements from email/BBS messages. http://www.amsat.org

Magic - X Window IC layout editor. [14]
http://www.research.digital.com/wrl/projects/magic/magic.html

Sigview - X Window graphical signal viewer. [14]
ftp://ftp.sunsite.edu/pub/Linux/apps/circuits/sigview31.tgz

Acrobat Reader - X Window displays and/or prints a file in the .pdf format.
http://www.adobe.com

Netscape - X Window web browser.
http://www.netscape.com

Mosaic - X Window web browser.
ftp://ftp.ncsa.edu/Mosaic/Unix/binaries/
http://www.ncsa.uinc.edu/SDG/Software/xmosaic

Conclusion.

X Window & Linux is for those hams who have a background in network operating systems. Windows 3.1 & MS-DOS is for the others, since installing them and their associated software is easier. The author has spent no money on software

(excluding the Slackware freeware distribution cd-rom [10]) since he started using Linux, but has spent many joyful hours configuring for the cd-rom drive, sound card, modem, etc., and installing application freeware. This is not to say he will not buy application shareware and commericalware in the future for for his Linux machine.

Acknowledgement.

The author is grateful to Len Warren, NOQHZ, a MS Windows 95 user, for his technical review of a draft.

Ellen Bandy pointed out grammatical errors for which the author is thankful.

References.

[1] Jeff Tranter, VE3ICH, "Packet Radio Under Linux", Linux Journal, SSC, Seattle, WA, Issue # 4 1, pp. 44-47, September 1997, phone 206-782-7733.

[2] Terry Dawson, VK2KTJ, "Linux AX25-HOWTO", ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/AX25-HOWTO, vl.4, 2 March 1997.

[3] Cort Dougan, "Native Linux on the PowerPC", Linux Journal, SSC, Seattle, WA, Issue # 37, pp. 43-45, May 1997, phone 206-782-7733.

[4] Fred Treasure, "NF/Observatory Networking with Linux OS", Linux Journal, SSC, Seattle, WA, Issue # 34, pp. 14-17, February 1997, phone 206-782-7733.

[5] Sebastian Kuzminsky, "Linux Out of the Real World", Linux Journal, SSC, Seattle, WA, Issue # 39, pp. 32-37, July 1997, phone 206-782-7733.

[6] Terry Dawson, VK2KTJ, "Linux HAM-HOWTO, Amateur Radio", http://sunsite.unc.edu/pub/Linux/docs/HOWTO/HAM-HOWTO, v2.3, I April 1997.

[7] Randy1 Britten, "Book Review - Programming with GNU Software", Linux Journal, SSC, Seattle, WA, Issue # 38, pp. 69-71, June 1997, phone 206-782-7733.

[8] Linux Journal, SSC, Seattle, WA, Issue # 38, pp. 4 & 17, June 1997, phone 206-782-7733.

[9] Patrick Reijnen, "Linux Hardware Compatibility HOWTO", http://sunsite.unc.edu/pub/Linux/docs/HOWTO/Hardware-HOWTO, v97.2, 14 June 1997.

[IO] Bryan Phillippe and Linux Journal Staff, "Linux Distributions Compared", Linux Journal, SSC, Seattle, WA, Issue # 29, pp. 20-33 & 45, Sept. 1996, phone 206-782-7733.

[11] Richard Parry, W9IF, "XNET - A Graphical Look At Packet Radio Networks", Proceedings of 15th ARRL and TAPR Digital Communications Conference, American Radio Relay League, Inc., Newington, Ct, pp. 64-75, 1996, phone 1-888-277-5289.

[12] Bryan Phillippe and Linux Journal Staff, "Linux Distributions Compared", Linux Journal, SSC, Seattle, WA, Issue # 29, pp. 20-33 & 45, Sept. 1996, phone 206-782-7733.

[13] Kevin Cosgrove, P.E., "Analyzing Circuits with SPICE on Linux", Linux Journal, SSC, Seattle, WA, Issue # 39, pp. 16-25, July 1997, phone 206-782-7733.

[14] Toby Schaffer and Alan W. Glaser, "An Introduction to IC Design Under Linux", Linux Journal, SSC, Seattle, WA, Issue # 39, pp. 54-64, July 1997, phone 206-782-7733.

Trademarks.

No Warranty.

Amateur Radio on Manned Space Vehicles: Improving Amateur Radio's Future Through Enhanced Space Frequencies

Frank H. Bauer, KA3HDO
AMSAT-NA Vice President for Manned Space Programs

Abstract

Since 1983, Amateur Radio has had frequent or continuous presence on space vehicles with astronauts and cosmonauts on-board. To date, tens of thousands of amateur radio operators and their guests have communicated with astronauts and cosmonauts in space. Despite the outstanding success of this facet of amateur radio, it has been plagued with a significant problem- many parts of the world, including most of the U.S., cannot reliably receive the 2 meter signals from the spaceborne crew members due to severe frequency interference. This problem is even worse for our amateur radio colleagues in space. This paper intends to describe the problem that astronauts and cosmonauts in space and terrestrial amateur radio operators endure to achieve contact success. It also provides some high-level recommendations to relieve this problem in the future.

Introduction

Amateur radio on human-operated space vehicles started in 1983 when U.S. astronaut Owen Garriott, W5LFL, was granted permission by NASA to fly a 2 meter hand-held transceiver on the Space Shuttle Columbia. Since that first mission on STS-9, the Shuttle Amateur Radio Experiment (SAREX) has flown 24 times on all of NASA's Space Shuttle fleet. In 1986 the Russian Space Station Mir was launched. Shortly thereafter, amateur radio was installed on Mir. This was accomplished through joint cooperation by the German Space Amateur Funk Experiment (SAFEX) team, the Russian Mir Amateur Radio Experiment (MAREX) team and the U.S. Mir International amateur Radio Experiment (MIREX) team. Since these humble beginnings 14 years ago, amateur radio has become a mainstay on all Russian and U.S. space platforms and will continue this tradition permanently on the International Space Station (ISS).

On Earth, remote scientific and research outposts like Antarctica have used amateur radio to provide psychological solace for the members of the research team and educational opportunities for student groups. Like their Earth-bound researchers, the Shuttle and Mir astronauts and cosmonauts use amateur radio as a spontaneous communication tool to permit random communication with people on the ground and pre-scheduled contacts with their friends and family. Early on, the international teams who coordinate the SAREX, MIREX, SAFEX and MAREX programs recognized the high visibility and tremendous appeal this new facet of amateur radio offers the general community. As a result, all these teams have implemented educational programs using communications between astronauts and cosmonauts as a means to pique student's interest amateur radio, science and technology. These programs have been tremendously successful. They provide our international youth a stimulating pathway to begin the amateur radio hobby and provide an amateur radio experience to whole communities that is positive and remembered for a lifetime. These positive experiences are vital for the future of amateur radio. Today's student hams represent amateur radio's future. Moreover, the positive experience to the community is vital in an era when antenna covenants and radio frequency interference issues threaten the viability of ham radio's future.

When crew-operated amateur radio in space began in 1983, it was very difficult to select frequencies that would be compatible in all parts of the world. The 2-meter bandplan in IARU (International Amateur Radio Union) Region 2 (North and South America) is very different from what is used in Region 1 (Europe, Middle East and Africa) or in Region 3 (Asia and Australia). This problem has gotten significantly worse over the past 14 years due to the popularity of packet radio in the U.S. and the significant worldwide influx of new radio amateurs that have flooded the 2 meter band. Crowded frequencies requires frequency sharing and strict frequency coordination. These methods have worked reasonably well for most terrestrial-based hams; however, they have not for those who wish to communicate with the astronauts and cosmonauts. From an astronaut's perspective, this frequency problem makes the worst DX pileup look like child's play. The orbiting crews are, many times, quite

frustrated with the inability to communicate with their fellow hams because of unwanted frequency interference. The following sections describe the problems that the space communicators (hams on the ground and the crew on-board) face everyday and some potential solutions to the problem.

Communicating With Space Vehicles— Similarities and Differences with Traditional VHF Communications

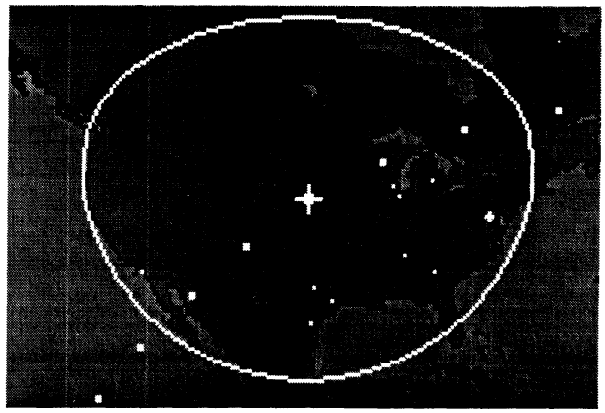
Before we delve into the question of frequencies, let's first understand how space travel effects amateur radio communications. There are three significant effects that space communicators experience which are vastly different from what a VHF or UHF ham radio operator traditionally experiences. These include 1) a significant change in station visibility, 2) the requirement to compensate for the Doppler effect and 3) the extremely long path length of the signals which results in weak signal communications.

Space Vehicle Visibilitv

VHF QSOs are predominantly accomplished using "ground-wave" (as compared to "sky-wave") communications techniques. Therefore, the contacts are usually line of sight. The higher your antenna, the further you can communicate. If you are driving in your car and operate simplex with another car, your communications "circle" is about 1-2 miles. If you increase your effective antenna height using a repeater, your communications "circle" increases to 15-30 miles or more. Space vehicles literally take the "repeater" idea to new heights. Figure 1 illustrates this effect quite clearly for the Russian space station Mir. As shown, the visibility circle encompasses the entire continental U.S. at times. The white dots that traverse from the bottom left of the picture to the upper right represent the motion of the center of this visibility circle every two minutes. Thus, the center of the visibility circle moves from around New Mexico to Wisconsin in about 6 minutes.

Figure 1 provides a graphical representation of several points that are crucial to understand the frequency issues

I. Vehicles in space see very large parts of the world, providing a great communications device
2. Space vehicles move quite fast over a terrestrial ham's station. Shuttle and Mir provide a maximum of an 8-10 minute communications opportunity for a ham during an orbital pass.
3. Due to their vantage point, space stations have "big ears." In other words, radio transmissions not intended for the astronauts or cosmonauts that occur on the space station uplink frequency will cause interference on the space station.
4. There are no borders in space. Figure 1 clearly illustrates that at one point in the orbit, Mexico, the U.S. and Canada can all communicate with Mir at the same time.



Space Station Mir Visibility Circle
During a North America Pass
Figure 1

Doppler Effects

The Doppler effect is the change in frequency that is observed by an individual when an object travels towards or away from that observer. When you stand near the track of a fast moving train, the whistle is high pitched as it approaches and becomes lower pitch when it passes by. Space stations move at 7.5 km/sec; so the Doppler effect is much more pronounced. A ground observer will see the Mir or Shuttle 2-meter downlink frequency increase up to a 3.5 kHz from its nominal frequency as the vehicle approaches. At closest approach, the downlink will be centered at the nominal frequency. As the vehicle moves away from the ground station, the observer will see up to a 3.5 kHz decrease in frequency from the nominal due to Doppler.

Doppler becomes important because it means that space vehicles need a wider channel separation as compared to ground-based activity. Currently, the FM channel spacing in the U.S. is either 15 kHz or 20 kHz. To guarantee interference does not occur with space vehicles, an additional 5- 10 kHz of separation is required on 2 meters due to the Doppler effect.

Long path length

Most VHF line-of-sight contacts are conducted with point-to-point path lengths no longer than 30 miles. Contrast this path length with 300 miles at closest approach for Mir and Shuttle. As figure 1 depicts, the Shuttle and Mir range circle is about 2500 miles in diameter (the width of the continental U.S.) This very long line-of-sight path length puts communications with these space faring vehicles in the weak signal category.

Despite these observations, there are times when hams on the ground have copied both Mir and Shuttle using handhelds tranceivers. While this reception is quite exciting for the ground-based ham, it rarely lasts for more than 30 seconds to one minute. Also, it usually occurs only when the space station attitude is favorable and while the vehicle is making its closest approach to the ground station.

To have a meaningful (>1 minute) conversation with the orbiting crew requires the use of receiver pre-amps and circularly polarized gained antennas. Strong terrestrial signals close to the Shuttle or Mir downlink will make reliable communication with the space station untenable due to the spillover of signals through the pre-amp or due to ground station receiver desensitization. This issue is quite apparent on Mir where the current space station downlink (145.80 MHz) is within 10 kHz of the APRS frequency (145.79 MHz). Strong terrestrial FM operations adjacent to weak signal space operations is detrimental to effective space communications.

Astronaut and Cosmonaut Experience

Many of the astronauts and cosmonauts who are hams are not your "dyed in the wool" radio amateurs. They are accustomed to using radios for space communications, but have rarely experienced a ham radio DX pileup or severe QRM. When faced with continual interference from voice repeaters, blasts from packet radio stations and stray voice snipets from simplex operators, the orbiting crew soon grows weary of ham radio as an effective communications medium. It is also very difficult for the orbit crew to change frequencies as they pass from one territory to the next. What are needed are clear uplink channels to the crew members and a set of frequencies that will not require the space crews to switch frequencies from one part of the globe to another.

Summary

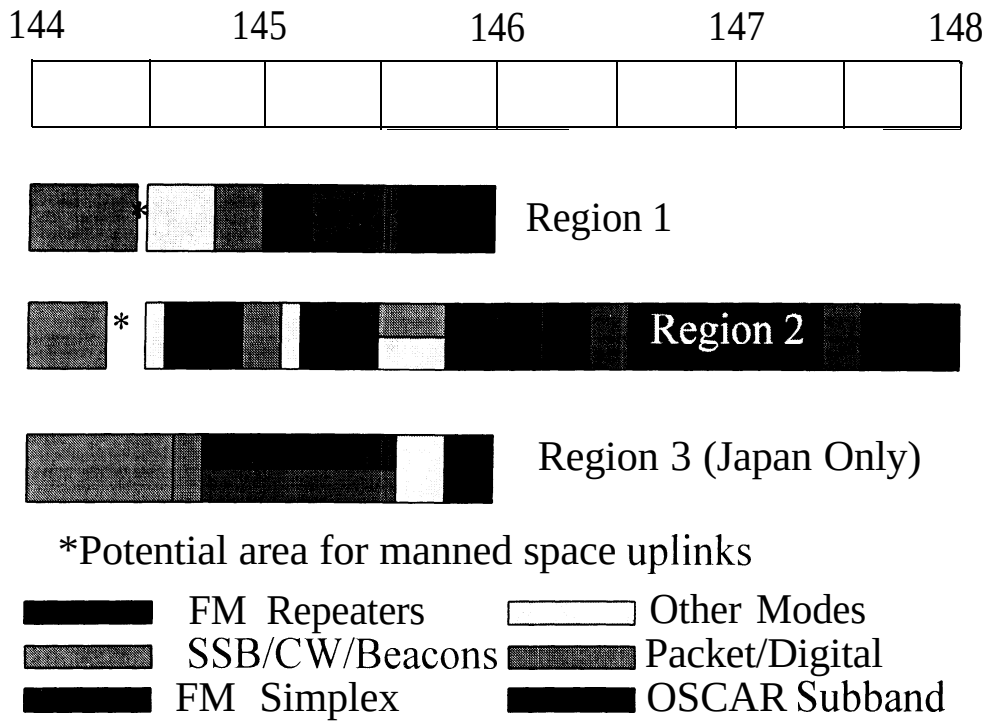
In summary, to effectively communicate with Shuttle, Mir and ISS crews using VHF requires:

1. Clear uplink and downlink frequencies.
2. A minimal channel separation from other activities on 2 meters of'at least 20 kHz with 25-30 kHz being preferable. This separation will cover the Doppler shifts as well as the weak signal concerns.
3. Frequencies that can be used throughout the U.S. since the space station's visibility encompasses the entire U.S. for periods of time.
4. Frequencies that can be used world wide since the space station overlaps several countries at the same time.

Frequencies in Space--What's the Problem??

Right now, frequency interference for manned space vehicles is a tremendous problem on 2-meters. The three IARU regions (Region 1, Region 2, and Region 3) each have differing bandplans. See figure 2. As shown, in many parts of the world the two meter band is only 2 MHz wide (144- 146). Since frequencies at VHF and above are primarily used for line of sight communications, these frequencies have been traditionally coordinated at the local level with no concern for global coordination. This means that many countries within an XARU region each have differing bandplans or "gentleman's agreements". This issue is even worse in the U.S. where "local coordination" occurs at the city, territory (e.g. Southern California, Mid-Atlantic, etc.) or state. In space, this "local coordination" becomes a problem because line of sight communications on the Space Shuttle and Mir (and eventually the International Space Station) overlap several cities, countries or continents simultaneously. This causes interference in space and on the Earth and a violation of these gentlemen's agreements. To date, the 2 meter band represents the most challenging coordination effort because it is the most used amateur radio band and it is currently the primary band for SAREX and Mir.

Until last year, the Mir crew used 145.55 MHz simplex as the amateur radio 2-meter frequency for voice and packet. This frequency was also used as a downlink frequency for SAREX. Many international organizations, especially the European community, have asked that Mir and SAREX move from the 145.55 MHz frequency since it is a popular simplex frequency. See figure 2.



2 meter (144 MHz-148 MHz) Bandplans for IARU regions 1,2, & 3
Figure 2

The Mir crew are currently using 145.80 (downlink) and 145.20 (uplink) for voice and packet. These changes were made by the Russian MAREX team and the German SAFEX team to conform with some of the manned space frequency recommendations that came out of the 1996 Region 1 (Europe, Africa and Middle East) IARU conference in Tel Aviv, Israel. It should be noted that these frequency recommendations have not been approved by the other two IARU regions. While this specific frequency recommendation may work well in parts of Europe, it violates many of the bandplans utilized in Region 2 and Region 3. In particular, 145.20 is absolutely untenable in the U.S. since over 140 repeaters in this country use this frequency or frequencies within 10 kHz of this frequency. Therefore, since the changeover, many U.S. radio amateurs who have attempted to contact Mir have been cited by other local radio amateurs for not following the Region 2 bandplan. This change has also resulted in considerable repeater-generated QRM on-board Mir. This complaint has been lodged by the astronauts and cosmonauts who use the radio on Mir.

The use of 145.80 as a manned space downlink is also a major problem. The primary issue in the U.S. is that this downlink is very near the APRS frequency of 145.79. The primary rationale behind the use of 145.80 as a downlink frequency is that it is right at the edge of the weak signal OSCAR sub-band. This frequency choice is considered to be an excellent compromise as a "guard" between the weak signal satellite users and the terrestrial VHF hams. As stated previously, the Mir and Shuttle downlinks are considered weak signal FM operations. The AMSAT international community would like to keep FM manned space downlinks at or near the OSCAR sub-band edge to minimize interference with CW/SSB weak signal satellites like AMSAT-OSCAR 10 and eventually Phase 3D.

The 145.80/145.20 pair used to be a repeater frequency pair in Europe. It should be noted that the European VHF societies mounted a great campaign over many years to move repeaters off this frequency pair. This was accomplished because 145.80 is right on the band edge of the OSCAR sub-band and these repeaters were interfering with satellite operations. Now that the 145.80 frequency is clear, the European VHF society believes using this frequency is an excellent choice for Mir, Shuttle, and ISS in Europe and will provide and effective way of keeping VHF repeaters in Europe from re-establishing this frequency pair.

In reviewing figure 2, one might arrive at a solution to move the manned space activity into the OSCAR subband (145.80- 146). While the Mir, Shuttle and ISS downlinks are considered weak FM signals, uplinks from terrestrial based hams clearly are not. The AMSAT international community is extremely concerned that high powered uplinks in the weak signal OSCAR sub-band will cause severe interference to OSCAR-l 0 and eventually to the sensitive receive systems on Phase 3D. The compromise is to use frequencies on the sub-band edge (145.80) or close to the sub-band edge for downlinks and move the high powered uplinks to an area well away from the OSCAR sub-band. As shown in figure 2, the asterisk (*) portion of the Region 1 and Region 2 bandplan provides an excellent area for potential manned space uplinks. A portion of this area in Region 2 includes the frequency 144.39. This may be an excellent frequency to move the APRS activities since part of Region 2 (Canada) uses this frequency for APRS now. A combined movement of APRS and the establishment of dedicated, world-wide 2-meter frequencies for Mir, SAREX and ISS will provide an unprecedented level of collaboration and compromise in amateur radio at the national and international level.

Manned Space Frequency Suggestions

The following manned space frequency suggestions have been presented to the AMSAT-NA/ARRL team as well as several IARU consultants in the US and Europe. These seem to solve the manned space frequency problems described in this paper and represent the best compromise between the satellite users and the VHF community.

Manned Space Frequency Suggestions:

1) Worldwide 2-meter Downlink Frequencies for Mir, Shuttle, and ISS:

   145.80, 145.8125* and 145.990* MHz

   *Backups or alternatives to primary 145.80 frequency

2) Worldwide 2-meter Uplink Frequencies for Mir, Shuttle, and ISS:

   144.490, 144.470 and 144.450 MHz

3) If a 600 kHz split pair is desired for Region 1 (Europe, Middle East, and Africa), the following is suggested:

Downlink	145.80
Uplink	145.20

4) The AMSAT-NA VP. for Manned Space Programs will work with the IARU, the ARRL and the U.S. Digital community in an effort to globally coordinate the above frequencies for manned space operations. Global coordination of all non confidential manned space frequencies for 15 meters, 10 meters and 70 cm is highly recommended and should be initiated as soon as possible.

Note that the above split mode frequency recommendations do not preclude simplex operations, if required. F'or simplex operations, the team will use frequencies which will minimize frequency contention such as 144.49 and 144.47, and 144.45.

Conclusions

Communicating with astronauts and cosmonauts is an exciting and challenging facet of amateur radio. Currently the orbiting crews and the ground-based radio amateur endure significant frequency interference issues to achieve success. These frequency problems have limited the growth and success of this communication medium. Moreover, the full potential of this facet of amateur radio to infuse new blood into the hobby through educational opportunities for students and its positive experience to the community has been somewhat stunted due to these frequency problems. Several suggestions have been made to improve the frequency issue on the Mir and the Shuttle. Let's take this opportunity to develop a compromise solution that benefits all and guarantees a strong future for amateur radio. Once accomplished, we can proceed with the design of the amateur radio station on the International Space Station with renewed vigor; knowing that it will soon become the ultimate station for experimenters, DXers and amateur radio educational outreach.

Proposal: An AMSAT Mobile TRAKNET

by Bob Bruninga, **WB4APR,**
wb4apr@amsat.org

Commercially, there are two distinct advantages of the global nature of satellites which cannot be easily met with terrestrial systems: wide-bandwidth point-to-point and mobile applications. With the availability of telephone, cable, and the Internet to link amateurs at fixed sites to each other routinely, we are wasting a lot of potential of our very valuable Amateur Radio satellite resources by ignoring mobile applications.

Ham radio is on the move. Any survey will likely show that many amateurs only have time to play radio while mobile, and similarly, whenever a ham contemplates a long trip, his ham radio is high on the packing list. Although many dream of taking along an I-IF mobile to play with and to report their progress back home, the $1000 to $2000 investment is just too much of a risk. Two meters is fun, and can bring emergency aid, but it just doesn't provide the nationwide coverage that is needed for the mobile ham traveler far from home, the offshore boater, or first-response teams arriving in a disaster area. In many cases, just a brief position/status report is all that is needed to assure the health and welfare of the traveler or to summon assistance or alert other communications channels.

Many hams have put together impressive mobile satellite stations, but the performance is poor and takes a major investment in dollars, size and weight. The reason is that they are essentially duplicating a typical weak-signal home satellite station on wheels. What we need is a new perspective which takes advantage of some very unique capabilities to exploit a small portion of our satellite on-orbit capacity to the mobile requirement. Fortunately, there are several Amateur Radio satellites that are ***very*** easy to transmit to from the mobile using only a 2 meter 25 watt FM mobile radio - the ubiquitous radio that everyone has.

Uplink

Lets look at the 1200 baud Pacsat uplinks. These uplinks are unique for several reasons that make them ideal for the mobile environment:

- The 2-meter uplink from a mobile omni antenna has a 9 dB advantage over a similar 435 MHz link due to the three times larger antenna aperture.
- There is no tuning or tracking required on the uplink since the Doppler on 2-meter is less than 3 kHz.
- Any 25 watt mobile 2-meter FM rig can be used as the transmitter.
- Any TAPR-2 compatible TNC can be modified for the uplink for $2.
- World-wide coverage.
- No software or hardware on-orbit modifications to the satellite.

Reportedly, stations running as low as seven watts into an indoor omni antenna have reported success with the 1200 baud PACSATS. This means that even backpackers with an HT and handheld gain antenna could get emergency or priority traffic into a Pacsat from anywhere on Earth! Figure 1 shows the results of the SPRE experiment during STS-72 when there was a digipeater on the Shuttle for station position/status reporting.

Downlink

OK, so the 2-meter ***uplink*** is easy and anyone can do it, so what about the downlink? This is not so easy. The path loss omni-to-omni is 9 dB worse, the satellite is only transmitting a watt or so for another 13 dB worse performance, plus it ***requires*** Doppler tuning, a $250 PACSAT modem and a $1000 all mode UHF receiver! In most cases, all successful Pacsat stations use high gain antennas and automatic tracking to make up for the more than 22 dB performance difference on the downlink. This is not something that most operators will want to add to their mobile. But what if the mobile application did not need to receive data, but only send it?

Traknet

The combination of easy uplinks, ***minimum*** downlinks, and an application that often only needs a one way exchange of data, such as the mobile position/status report is the whole idea behind Traknet. Only a few automated downlinks are needed every 1000 miles or so to receive the mobile data and to provide it into a nationwide system of linked ground stations. These ground stations relay the mobile position/status reports onto local mobile vehicle tracking channels and onto the Internet. Anyone may access these data live on VHF, HF or via the Internet. Traknet is not just a future idea, it can be implemented immediately with existing equipment and satellites. Yes, even the

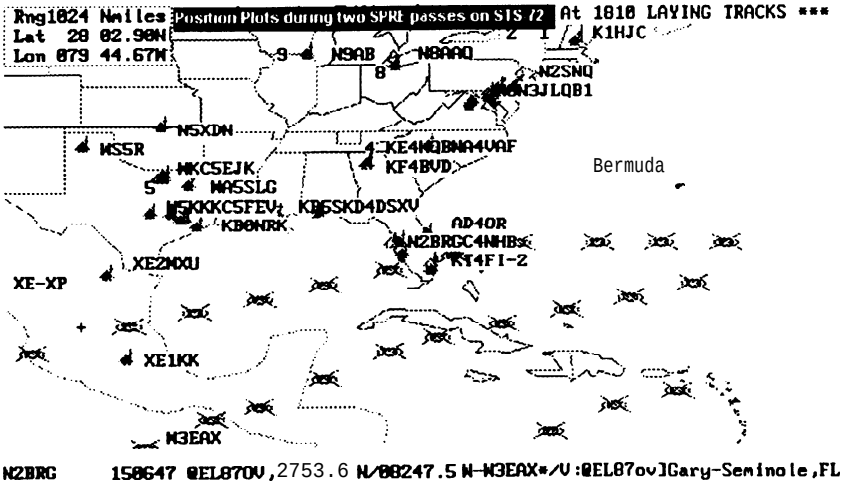


Figure 1. This figure shows the display of stations in the US that successfully digipeated their positions and status via the SPRE experiment on STS-72. A similar experiment was conducted via SAREX on STS-78.

Internet ground stations exist. Already there are six Traknet-type stations on the World-Wide-Web. Just link to any of the following sites and you will see *LIVE* (or nearly live) fixed and mobile amateur radio position/status reports.

- ATLANTA: *http://www.wadsy.radio.org/aprs/index. html*
- CALIFORNIA: *http://sboyle.slip.netcorn. com/LIDSAPRS. html*
- CHICAGO: *http://tbcnet.com/~jleonard/noiltest. html*
- MIAMI: *http://www.bridge.net/~sdimse/javAPRS.html*
- ONTARIO, CANADA: *http://www.peel. com/javAPRS. Html*
- WASHINGTON, DC: *http://web.usna.navy.mil/~bruninga/aprs.html*

You only need to access one, since most of them have links to each other, and more are coming on line monthly. Some of these sites are already listening to many such status/position reporting channels. But the problem is that none of these sites is yet listening to the Pacsats mostly because setting up an automatic Pacsat ground station is not trivial and the guys who play with the Web all day are not the same guys that are necessarily fully invested in Pacsat hardware. All we need are probably eight stations scattered over the continental USA to implement a reasonable Traknet system as shown in Figure 2.

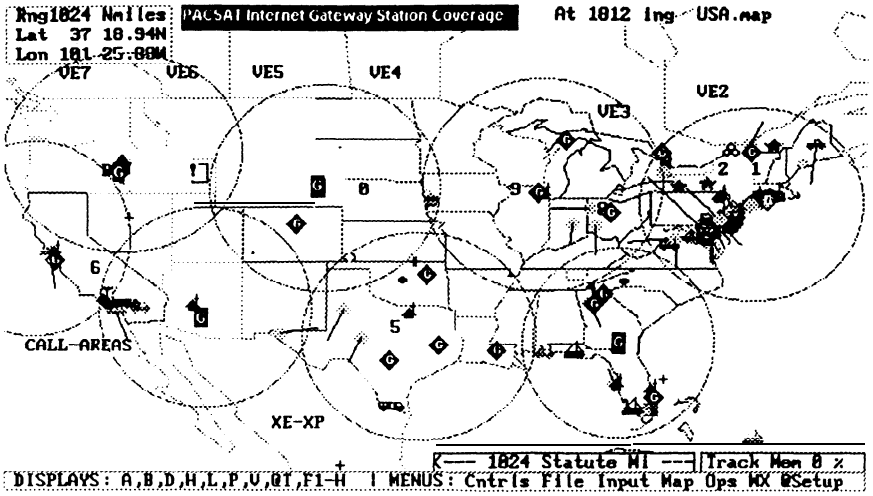


Figure 2. This figure shows how eight reasonably located PACSAT Internet Gateway stations could provide nationwide coverage for mobile travelers. These range circles are a very conservative 600 mile radius or less. Two hundred or more mobiles could be accommodated per footprint per pass.

Mobile Station

A mobile station consists of nothing more than a typical 2-meter FM radio and a modified TAPR-2 compatible TNC as shown in Figure 3. Optional accessories are a GPS for moving position reports, and a laptop for entering messages. Most modem TNCs will accept the GPS data directly and will transmit the data in a timed packet burst. There is even a tiny handheld TNCs called the APRS Mic-Encoder that includes front panel switches for selecting 1 of 7 pre-canned status messages without needing a laptop to change the status report. The modifications to the TAPR-2 TNC are to simply ***exclusive*** or the transmit data with its 1200 Hz clock and to filter the result to the Mic input of the radio. The following circuit will do this with nothing but an 89 cent standard 7400 quad 2 input NAND gate connected as an XOR gate to the two points shown.

The pin numbers shown in Figure 4 are for a PacComm TINY-2.

Traknet Protocol

The problem with any Amateur Radio satellite is the very low bandwidth available compared to the very large worldwide Amateur Radio population. At first glance, the prospect of increasing the number of users on a Pacsat channel by a hundred fold raises lots of red flags in the minds of those stations who already find ten minutes of a satellite pass to be too short for any meaningful dialog. But what if each of these hundreds of new users was limited to only a few seconds per orbit? Then as many as 200 stations per footprint could be tracked. That is the only objective of the Traknet protocol, to allow everyone to transmit a few single l second position/status reports during the closest point of approach over their location. If only one channel is designated for Traknet,

Figure 3. This photo shows a typical PACSAT mobile status/position reporting station. It consists of a 2 meter FM radio, a GPS unit and a slightly modified TAPR-2 compatible TNC.

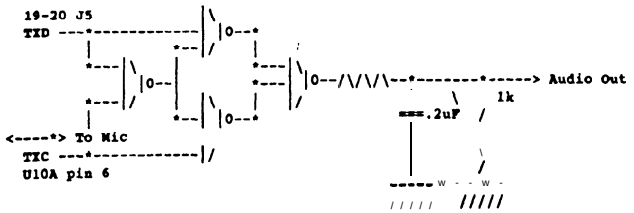


Figure 4. Pin numbers for a PacComm *TINY* 2.

then the other 3 channels are free for normal Pacsat use and no amount of congestion on the Traknet channel can interfere with existing users on the other three.

Traknet Satellites

There are currently five 1200 baud Pacsats on orbit. One, WO- 18 has actively invited UI frame digipeating and leaves DIGIPEAT on most of the time. The problem is that the WO-18 downlink is difficult to receive unattended due to a spur tone in the middle of the data which makes receiver lock a difficult and manual process. AO- 16 has had its digipeater ON for the last six months. Other Pacsats occasionally have DIGIPEAT turned on, but there is no formal policy. The purpose of this article is to encourage the designation of one good channel as a gathering point for Traknet experiments, and then progress can be made and the potential of Traknet can be evaluated. Here are the frequency plans of the existing Pacsats:

Digipeater	Uplink (MHz)	Downlink (MHz)
AO-16	145.860, 145.900, 145.920, 145.940	437.051
LU-19	145.840, 145.860, 145.880, 145.900	437.153
WO-18	145 .900	437.104
IO-26	145.875, 145.900, 145.925, 145.950	435.822

Advanced Mobiles

While the preceding was written to emphasize the ease of using the Pacsats by anyone for emergency or priority status/position reporting, there is certainly no reason why a full two way Pacsat communications system cannot be added to most mobiles. Omni Pacsat downlinks are possible and the addition of only a modest gain antenna will certainly help. Advantages are the small size of a 6 dB two element UHF antenna and the SHORT cable run found in a mobile. Rather than a $1000 SSB rig, a $90 QRP HF rig and a UHF downconverter could do just as well.

Conclusion

The advent of the handheld GPS unit for under $199 has brought thousands of mobile amateur radio operators into the world of mobile data. For years, the growth of amateur GPS applications have been growing at phenomenal rates. At this writing there are mobile map packages available which ***include*** the GPS unit for under $150 total! Similarly, the state-of-the-art in automatic Pacsat ground station technology has been improving with many recent software packages to make un-attended automatic ground station operation quite easy. The problem is that these two communities of expertise have so far had little cross-interests. It seems that the time is now to merge these technologies into a new amateur application that takes advantage of the unique capabilities of each and fuels the development of an Amateur Radio Mobile Satellite System. Traknet is the opportunity to not only merge these interests into a common purpose, but also to demonstrate Amateur Radio's continuing progress in communications technology. ■

APRS VISION SYSTEM

Bob Bruninga, WB4APR
115 Old Farm Court
Glen Burnie, MD 21060

The APRS Vision System (AVS) was developed to provide a variable bandwidth vision capability for APRS Robotic applications. The system provides an effecient method for image transmission from a mobile or rover and uses the standard APRS UI frame protocol so that the existing APRS digipeater networks can be used for vastly extended range. Using the APRS UI broadcast protocol, not only is there no wasted bandwidth for ACKS, but everyone can monitor the image. A hypothetical idea of driving a robot in New Jersey from a HAM shack in Maryland presents the concept which was so markedly demonstrated this year with the Mars Rover. The whole world watched as this tiny robot maneuvered around on the red planet. It would take an image, and then take minutes to hours to send that image back to earth, where the operators would determine their next moves.

A typical rover design would actually use full fast-scan NTSC video via an amateur television link for nearby line of sight remote control applications. But having the APRS vision system kick in when the ATV link fades would allow it to wander several orders of magnitude further away while still guaranteing a picture for control purposes. A hazardous duty rover could thus penetrate much further into a building, or in the remotest areas, and still have a viable means of visiual navigation.

Although the vision system is intentionally limited in resolution to fit the available APRS channels, it does provide a reasonable vision capability for remote robotics applications with the following specification:

- 16 level VGA compatible gray scale
- 1x2 aspect ratio
- Full picture variable resolution up to 128 by 256 Pixels

The gray scale was limited to 16 levels to reduce the transmission bandwidth and because of the limitation of VGA displays to only 16 shades of gray. The 1x2 aspect ratio was selected to maximize the wide horizontal navigational view while minimizing wasted bandwidth in the vertical. An option could allow joining two images to give a 256x256 view if needed. But the most appealing feature of the system is the variable bandwidth algorithm. This assures that only the minimum necessary picture bandwidth is transmitted, but the final resolution is under the control of the receiving station!

The variable bandwidth simply means that the first packet contains the complete image, although at very low resolution. For each doubling of the number of packets, there is a doubling

of resolution. This allows the user to decide when he has enough resolution to proceed with his next step. In some cases, the single full image packet may be sufficient to allow the control operator to make the next navigation decision. If not, then he waits for 3 more packets and sees double the resolution. Then 12 packets, then 48, and finally 192 if he wants the full 128 by 256 pixel resolution. At any time the receiving station is satisfied with the image, he can send a QUIT message to stop further transmission. The results of such a variable resolution scheme are tabulated below:

LEVEL	PACKET	IMAGE RES	DISPLAY SIZE	TOTAL TIME	TIME DIGIPEATED
1	1 st	8 x 16	quarter	1 sec	3 sec
2	2-4th	16 x 32	quarter	4 sec	12 sec
3	5-16th	32 x 64	half	16 sec	48 sec
4	17-64th	64 x 128	half	1 min	3 min
5	65-19nd	128 x 256	full	4 min	12 min

Notice that on the receive end, the poor resolution of the lower images are somewhat masked by being presented in a smaller display. By showing the poorer images at one quarter of full size, the crudeness of the rough pixels is somewhat mitigated. These imgages are presented in a zoom window which expands as the resolution increases as shown in Figures 1 through 5.

Although the image in the first packet is very crude, a good example of its usefulness is a picture of a hallway where the rover is trying to decide when it has reached a turning point In this case the operator may only need to preceive a solid color door on a white wall to make his next navigation decision. In this case, the first single packet may be sufficient. Not only does this reduce channel bandwidth by two orders of magnitude, it can also be thought of as giving a 100 fold improvement in image speed. If you can navigate on a single packet, then you can navigate at a 1 second update rate! If you need a better image, however, then your refresh time is much slower and you have to wait.

PROTOCOL

The individual packets can be thought of as arrays of pixels with dimensions as shown Each array represents an 8 by 16 grid of boxes where each element in the array is the average value of all pixels in that box. The next array contains a similar average of all pixels in each quadrant of each box. Notice that one quadrant in each successive array can be calculated as the difference between its other three quadrants and the previous array.

R1(x,y	first packet
R2(3,x,y	next 3 packets
R3(4,3,x,y	next 12 packets
R4(4,4,3,x,y	next 48 packets
R5(4,4,4,3,x,y	next 192 packets

Since the array indicies are transmitted along with the packet, not only does APRS knows how to process them, but also in the case of any missed packets, the receiver can request a repeat of the missing packet.

TRANSMISSION

Normally, for real time applications, the image will be transmitted only once. But since APRS UI frames are used, some redundancy is required for the critical packets. Since the packets in each level are just a doubling of resolution over the previous level, individual packets become less and less critical to the final image. For this reason, the first packet is transmitted like any other APRS packet on a decaying time period. Like other APRS packets, once it gets down to every 10 minutes direct, or every 20 minutes via one digipeater, it will continue at that rate providing a beacon that such an image is available for anyone tuning in late.

The next three packets are transmitted on the same algorithm, but only 3 times each. Then they stop unless individually requested. The next 12 packets are transmitted only twice and all other packets are transmitted only once. Over the full image, this transmission redundancy only adds about 13% channel load.

If for some applications, continuous broadcast of an image is needed, then the image packets can be transmitted at some very low rate in the background continuously. Im not sure of the application, but if 10% of channel capacity through one digipeater was used for image transmission, then a new level 4 image could be delivered about once every 30 minutes or a level 5 image once every 2 hours. I'm not sure of the application or value of this technique, however...

WARNING

Although this system is designed for on the air use with the existing APRS networks, it is absolutely *not* intended as a means for the exchange of FAX images between fixed stations on a routine basis. The APRS bandwidth just cannot support it. It should ONLY be used in specific high priority robotic or remote vision applications.

Figure 1. The original image as published in *QST* page 11 August 1993.

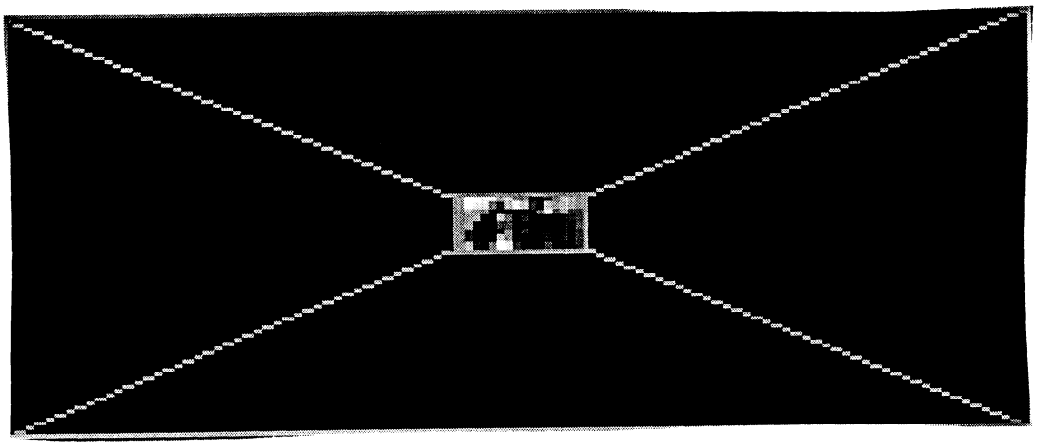


Figure 2. The first packet image at Level 1. Shown at one quarter size. It is 16x8 and 16 shades of gray.

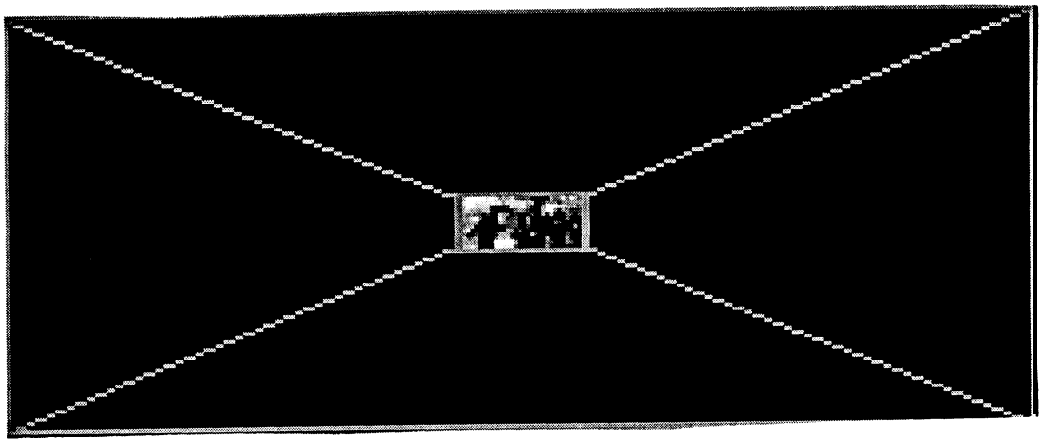


Figure 3. After 3 more packets the image is 32x16 and 16 shades.

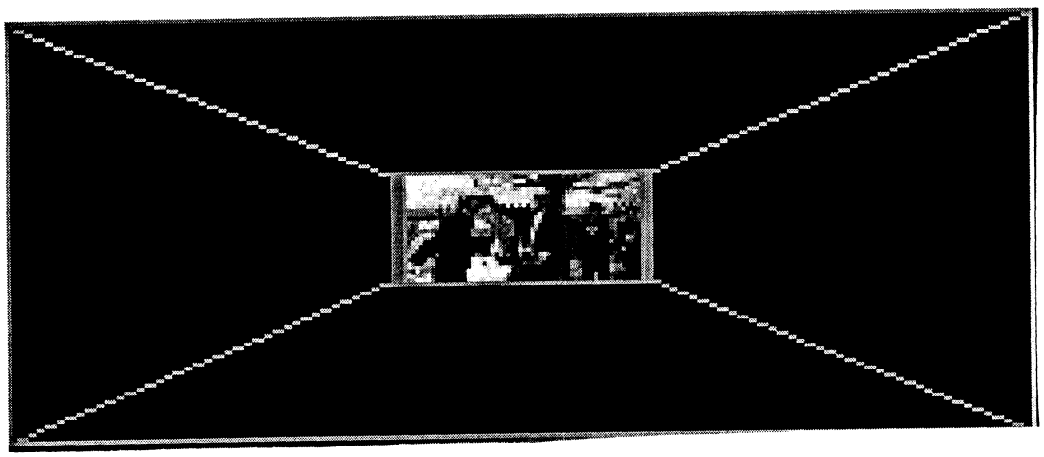


Figure 4. After 12 more packets the image is 64x32. After 48 more packets the image resolution will again double to a level 4 image (not shown).

Figure 5. After 192 more packets, the full image of 256x128 pixels is displayed at full size and 16 levels of gray.

APRServe: An Internet Backbone for APRS

Steve Dimse K4HG
http://www.aprs.net/k4hg.html
k4hg@tapr.org

Last year at the 1996 Digital Communications Conference I predicted that within the next year we would have a working nation-wide APRS backbone running on the internet. This paper details the progress that. has been made towards that goal.

At the time this is written (early August 1997), there is full time live data available from San Francisco, Los Angeles, Nashville, Atlanta, Miami, and New Jersey. Several other sites offer continuously updated files containing almost-live data.

Each of these sites functions independently, using a local WWW server to display the data with javAPRS. The first step has been taken to link these sites into a seamless network with a program I have written called APRServe.

In planning APRServe, I had several goals in mind.

1. The program should relay data from a local TNC onto the internet, just as the other Internet. Gates (IGates) do.

2. The program should be able to connect, to a list of other servers to obtain live data from other full time sites, and to interconnect with other copies of itself running in other locations.

3. There should be an echo function to allow non-permanent. sources to connect to APRServe, and add their data to the stream.

4. The data coming in from the various sources should be filtered to eliminate as much redundant data as possible.

5. There should be a buffer of data to allow the immediate transmission of all known information to a newly connected client.

6. There should to be a way to remotely monitor the status of the server.

7. Local TNC data should be accessible either alone or as part of the merged stream of all data.

Implementation

When I first envisioned this project, I intended to reuse much of t.he Java code I had written for javAPRS. However, as I worked more with Java I came to realize that it is not yet mature enough to handle the demands of a full time server application. I have written the program in C++ under MacOS 7.6, using Apple's Open Transport networking architecture for peak performance. The resulting program is much faster and more stable than a Java program

would have been, but is not transportable to other computer systems. However, while writing the program I made every effort to make it as easy to port. as possible. This means no fancy user interface, parameters are read from text files when the program initializes, and status messages are displayed in a simple text window.

Networking

On startup, the program reads a file containing a list of full tim; data servers, and connects to them. It monitors each connection, and if no data is seen in 15 minutes, the connection is assumed to be dead, and a reconnection is attempted every 15 minutes until successful.

Multiple copies of APRServe can interconnect, sharing data among themselves. When other full-time servers are available, a failure in one will not mean that the entire network becomes unavailable. Also, the loads of both clients and IGates can be shared between all operating servers. At present, the protocol between the servers is quite simple. The only difference between a client. port and an interconnect. port at present is that the data sent out to an interconnect port does not contain any data that arrived by that port. (Without this difference, data would echo back and forth between two sites.)

Data Echo

In an ideal world, there would be a full time Internet. gate (IGate) within a single digi hop of everywhere in the country. However, since a full time IGate requires a permanent connection to the internet, this coverage is unlikely t.o ever be attained. To fill in the gaps between full time IGates, individual users of Mac/WinAPRS will be able to send their TNC data to APRServe and have it relayed to other internet users. This feature can also be used for special events by having simple programs that send data from a TNC to APRServe.

Data Filtering

As anyone who has watched the raw data on a busy APRS net can attest, there is a lot of redundant. information in the stream. Others are working on improvements to the TNC digipeater code and protocol to reduce the redundancy. However, given that there may be a large number of redundant. data streams feeding into APRServe, the possibility exists to quickly overwhelm the average dial up internet connection.

To address this I have developed a simple filtering routine that has low processor overhead and good selectivity. As each string comes through APRServe, the program calculates a one byte bitwise checksum and a character count on the data portion of the packet (that which follows the ':', thereby excluding the callsign and digi info). The checksum and count are combined with a character count of the data into a 16 bit integer hash code. For each data stream the program maintains a buffer of a variable number (set at compile time) of packet callsigns and their hash code. The incoming packet's hash code is compared with those of the packets in the buffer. If a match is found, it is confirmed by checking the callsign at the start of the packet, and if confirmed the packet is discarded. If the packet is not in the buffer, then it. overwrites the oldest data in the buffer and the packet is passed on for processing.

The optimum size of the packet buffer is still to be determined. A larger buffer would cut out some longer period QRM, such as the parked vehicle tracker with the GPS turned off that repeats the same posit every minute. A small buffer, say 16 packets, cuts out all of the pings between digis (a big problem in Miami) with much less processor overhead, and is what I presently use.

In a test on 72 hours of data from the Miami LAN, this method was found to reduce the size of data bv about 60% with no information loss. Using a buffer of 128 packets only improved the reduction to 68%.

Data Buffering

APRServe presently hears somewhere between 450 and 600 stations, depending on propagation. However, many of these are heard infrequently, either because of distance to the IGate, or the low transmission rate used by fixed stations. To let a user see all the heard stations immediately upon connecting to APRServe, the programs keeps an internal buffer of heard stations. This buffer stores a single packet of each of three different types for each station: weather, position, and other. Each incoming packet is parsed to determine it's type, and it overwrites the prior packet. When a new client. connects, the contents of this buffer are sent immediately to the client.

In order to assure the timeliness of the data, every half hour the buffer is checked, and if no packet was heard from a station in the prior 12 hours, the station is purged from the buffer.

Remote monitoring

Since the APRServe machine runs in a location I cannot easily access, I have added a status port. When I connect to this port. the program reports the present status of the program, including the current connections. Status messages are reported to the screen display and to the status port. Some very simple commands can be sent to the program from this route as well.

Client connection

To access the data, a client connects to the server using the TCP/IP Telnet protocol. APRServe supports connections on two ports. Port 14579 (chosen for the APRS VHF frequency) send only the local data from the TNC input. This keeps users from "drinking from a firehose" when they only want to see Miami data. Port 1015 1 (the HF APRS frequency) serves the full data stream.

Presently there are two clients for the APRS Internet Network. javAPRS, being "born" to networking, is fully functional now. MacAPRS, thanks to Apple's Communication Toolbox, is able, with a public domain tool, to connect to the network now. It. does not. handle the full protocol at this time. WinAPRS is expected to catch up to MacAPRS once the Sprouls get a chance. I expect that most "hard core" APRS users will prefer the Mac/WinAPRS client, as it. is faster and has more features than javAPRS. javAPRS is more useful for casual users, and most especially for newcomers and non-hams, since it is automatically downloaded and run by the Web browser.

But is it Ham Radio?

I hear this a lot. Of course it is ham radio. Our hobby is about communicating and experimenting; this system does both.

Ham radio needs to embrace new technologies, and concentrate on those aspects that are unique. Let's face it., a 200 mile 1200 baud packet. conversation fades compared to an internet 33.6 kbaud video chat halfway around the world. If ham radio continues to draw in upon itself and celebrate past glory, it. will wither and die. We must show that we can utilize the unique aspects of amateur radio, and merge them with existing technology to produce novel communications systems.

Future plans

1. TIGER map relay. javAPRS has had the capability for the last year of using the TIGER map server from the census bureau. The problem is that the security rest.rict.ions of java prevent. the use of data from servers other than the one from which the program is run. The solution is to relay the map images through the same server that. the applet is loaded from. Steve Boyle (KD6WXD) has this working on his server, and I will be adding the capability to my system.

2. Intelligent routing of messages. I am strongly against. using the internet. for the wholesale routing of traffic between different RF nets. However, the intelligent routing of messages would greatly increase the usefulness of the Internet. backbone while only minimally increasing the traffic. This year's prediction is that next year I will be presenting a DCC paper detailing how this nationwide messaging system was designed and implemented.

3. Improved interconnection. At present, multiple sites running APRServe all need to be connected to the same IGates in order to be assured of access to the data should another APRServe site go offline. The creates unnecessary load on both the IGates and APRServe sites. I am developing a negotiation protocol to allow the servers to split. up the IGate load dynamically. In this way, each server maintains a list of the available IGat.es and APRServe programs. When an APRServe site goes offline, this is sensed by the remaining APRServe site(s), and the missing server's load is split between the remaining site(s).

Appendix A
Annotated Bibliography of APRS Internet Sites

http://www.aprs.net/usa.html The Miami APRServe page, with links to various documents describing the hardware and software in more detail.

ftp://ftp.tapr.org/pub/tapr/SIG/aprssig/files This is the primary FTP site for APRS files. The latest. version of the Mac, Windows, DOS, and Java versions of the program are available here, as well as t.he largest collection of maps for all platforms.

http://web.usna.navy.mil/~bruninga/aprs.html Bob Bruniga's ("The Father of APRS") site contains many pages of almost live data. A must see site to understand APRS capabilities,

http://www.aprs. net/javAPRS.html A series of pages containing demonstrations of javAPRS's capabilities, as well as links to most of the other javAPRS pages I know about.

http://www.aprs. net/javAPRSprog.html The official documentation for javAPRS for those people who are interested in adding javAPRS to their page.

http://www.aprs.net/find/ It. is possible to use the APRServe network to follow individual stations in their travels. A new feature in javAPRS filters out all except specified stations. The above URL lists those pages which I have created for individuals. For example, http://www.aprs.net/find/k4hg.html shows my various stations, and http://www.aprs. net/f ind/w7lus. html follows the exploits of long haul trucker and long time APRSer Peter Gross. If you are on APRS and would like a page here let me know.

http://www.kcaprs.org The Kansas City APRS Group's site is one of the more complete sites for APRS info.

http://aprs.rutgers.edu Run by the Sprouls, containing the most up-to-date documentation on Mac/WinAPRS, and also has an FTP server that contains latest Mac/WinAPRS programs and lots of maps.

KEYPAD INTERFACE LANGUAGE:

Digital Language Aids Hams and Others Too

by WØLIQ and K9LTL -- 7/97

SYNOPSIS

ARDS Project was proposed earlier in the Proceedings of the 12th ARRL Digital Communications Conference (held in Tampa, FL, in '93). During that time, I? & D was limited to experiments with Model 12. We renamed ARDS Project Computer Assisted Communication (CAC). This system evolved by experimenting with more models. Model 17 revealed compelling reasons why FCC's 97 part rules need changes that would permit hams to use Keypad interface language with digital signals. (FCC's present objectives would not be affected by minor changes.)

WHY SHOULD **DIGITAL SIGNALS BE ILLEGAL ON HAM BANDS?**

If a large group of literate people gathered but no one attending could understand people there, communications would be meaningless! Hams cope with pidgin languages despite evidence that computers can assist them. CAC system's Model 17 offers hams substantive evidence that FCC's 97 part rules need changes so hams could exchange digital signals. Ham bands are finite (a fact emphasized recently by David Sumner, K1ZZ in QST, 8/97). More hams could interact with less bandwidth and translate languages. This idea may seem too easy to be true about something so hard. Marvin Minsky learned why minds are fooled while working on artificial intelligence at MIT. In his book (THE SOCIETY OF MIND) he advises how to keep our minds alive:

> In general, we' re least aware of what our minds do best. . . . It's mainly when our other systems start to fail that we engage the special agencies involved with what we call "concious-ness." Accordingly, we're more aware of simple processes that don't work well than of complex ones that work flawlessly. This means that we cannot trust our offhand judgments about which of the things we do are simple, and which require complicated machinery. Most times, each portion of the mind can only sense how quitely the other portions do their jobs. M. Minsky ('85)

ARE HAMS AWARE THAT KEYPAD SIGNS ARE COMMON DENOMINATORS?

Keypads are common denominators for MATH operations used in 20 countries. Hams in these countries speak and write languages with Indo-European origins. Their alphanumerics are stored in computers under ASCII codes. ASCII codes were not listed in an order suitable for cross-referencing. ASCII codes can be reordered by "converging" print and computer technologies. Algorithms can make this possible.

LP (Language Processor) software is required to record and play-back language elements. LPs are analogous to WPs (word processors). A basic distinction is: WPs assist WRITING in different languages. LPs assist INTERACTING in different languages. LPs record languages so other interface devices (keyboards, keypads, TDDs etc.) can be controlled via language using keypad script. Computer hardware and operating systems are essential also. We promote CAC for Computer Assisted Communication as an expression for components controlled via Keypad Interface Language. CAC system enables hams to communicate in foreign languages via directing use of computer's memories.

CONVERGE **PRINT** AND COMPUTER **FILES (An Idea** Whose **Time Has** Come)

Word processors are ideal for organizing language items in print files (for fast-easy referencing during conversations). Linguistic forms can be assigned indexes after contents of print **files will have** been compiled. Next, language processor software is employed to catalog all items processed earlier into playback type software. Print files provide CAC system's users with identical listings of data stored in databases under files, topics, and indexed records.

Print and computer listings are correlated so that data will have identical listings with only minor exceptions **(exceptions** are noted in explanatory CAC system manuals). **In** effect, **print** and computer technologies become "converged' to make CAC system operational.

COMMUNICATE WITH **MARTIANS** VIA **KEYPAD INTERFACE** LANGUAGE

Keypad interface language features script printed on keypads of full sized computer keyboards in over 20 countries. These keys are also available on notebook computers. Index numbers from one to four digits are assigned to virtual records stored in computer memory. Records hold linguistic forms in sizes that range from alphanumerics to paragraphs. CAC system users display records randomly on monitors at conversational rates. Non numeric signs function as commands. Those are inputted with indexes **(without** spaces **entered** between characters as via pressing spacebars, etc.),

COMMAND FUNCTIONS OF NON **NUMERIC** KEYPAD **SCRIPT:**

.	Executes "data selected via indexes" and file changes.
–	Joins indexes and other commands before executions.
+	Switches processing modes from SPEL to DATA and vice versa. (Typewrite in SPEL mode; Process **stored data in DATA** mode.)
/	Switches between data files (prefix for file **code** numbers).
(	Synonym for Enter key; erases in DATA mode and is used also for line feeds when typing in SPEL [sign **means press Enter].**
*	Serves as prefix in SPEL mode (it means **use SPEL to** chat).

NOTES: Keypad script has English words for numbers and commands. Three countries substitute math symbols for * and / signs. Modal use examples were listed in exercises **that follow.**

TELEPHONE NUMBER **AND** PUSH BUTTON **DIALING ANALOGY**

Telephone users lookup names alphabetically to find telephone numbers and push numbers accordingly. This is a rather simple task even for children. Telephone callers are also asked to press numbers, when **calling, to** reach parties wanted or to hear recordings.

CAC users send indexes for the data stored in a receiving ham's computer memory. **Indexes** are executed by adding some commands. Hams can copy received **CW** or **voice** signals on computer keypads or keyboards. **CW** signals could be copied on keyboards by reading Morse signals deciphered on devices that have LCD type readouts, etc.

We have not tried direct **connections** between CAC signals received and computer interfaces **(as** when using **RTTY). That seems practical only** if transmission facilities were to be private and thus secure. **Ham experiments can be conducted** only by using off-the-air methods because FCC's 97 part **rules forbid the use of digital signals.**

TYPICAL FILE AND CYBERTEX EXAMPLES FOR ANALYSES

In the brief examples that follow, you will be able to analyze how data is listed in print files, how data in files can be equated with respect to its meanings, and how Cybertex strings can be written to reference and/or record typical interactions.

A. EXAMPLES OF PRINTED FILE LISTINGS

Space does not permit more than a few examples on how data items can be listed in print files and exchanged via digital signals. CAC signals can be sent as Morse or English words (listed in Manuals). Word "Cybertex" is CAC system's name for written, digital strings.

Indexes	English File Listings	Indexes	Spanish File Listings
121	Good morning!	121	Buenas dias!
131	Do you speak English?	131	¿Habla usted ingles?
195	My name is_	195	Mi llamo-

B. HOW INDEXES ABOVE CAN BE EQUATED, SENT, COPIED, AND DISPLAYED

Cybertex	From English File EN1	From Spanish File ES2
121.	Good morning!	Buenas dias!
131.	Do you speak English?	¿Habla usted ingles?
195.+	My name is_	Mi llamo-
ROY GA+		

C. HOW INDEXES ARE JOINED, FILES CHANGED, AND MODES SWITCHED

Cybertex String writing

121-10-131-/2-195-.+ROY+

Display of Cybertex strings after having been inputted:

Good morning!_Do you speak English?_Mi llamo_

ROY+

DATA PROCESSING DETAILS OF EXAMPLES ABOVE

Notice in example A. that English and Spanish file listings have identical index numbers. Thus when accessing either English or Spanish files, the same indexes would be inputted and executed as in example B, and meanings will be equivalent. Cybertex in C. has a longer string (it displays on two lines). A minus [-] sign is used to join indexes. This [/2] combination moves data processing within English file to processing it within Spanish file. Sign [+] changes DATA mode to SPEL mode for typewriting (spelling out) name "ROY."

Operators can monitor Cybertex inputs, and send signals to cancel or correct input mistakes. Cybertex inputs are displayed briefly. Next, strings become replaced by data referenced. Printer outputs and digital speech are optional (speech requires extra software).

APPLICATIONS FOR CAC SYSTEM AND OTHERS TOO

Keypad language enables people to cross-reference between human memory and computer memory. CAC system allows new kinds of interactions worldwide because Language processor tools can be programmed to translate over 20 languages. Linguistic forms in language can be organized, indexed, and accessed for conversations to interact in various contexts among which is the context of telecommunications. Standard ENCRYPTION KEYS are required by CAC users and governments. CAC system offers applications for persons who have handicaps also.

R.E. and M.S.

An All-software Advanced HF Modem for Amateur Radio[1]

Matthew Ettus, N2MJI[2]
Carnegie Mellon University
733 Maryland Avenue, Apt. #3
Pittsburgh, PA 15232
mne@cmu.edu
http://www.andrew.cmu.edu/user/mne/

Abstract

The need for an inexpensive and robust replacement for 300 baud FSK on the amateur HF bands is apparent. A modem was developed which allows for greatly increased data bandwidth (up to 500bps), while at the same time allowing for increased reliability through the use of advanced modulation and coding methods. The entire system runs on a standard PC with a soundcard, under Linux. The only necessary hardware is a mechanism for keying the radio.

Introduction

The main goals of this design were to create a modem which is suitable for use in the amateur HF bands, which would provide high throughput, and, at the same time maintain low error rates. Also important was making the system accessible to all amateurs without the purchase of additional hardware. The code had to be simple and efficient enough for the additional processor load to be unnoticeable to the user.

Most amateur radio HF modem designs have concentrated on confining transmitted signals to narrow bandwidths, usually 500 Hz or less. In order to obtain adequate throughput, higher order modulation schemes, such as 8– or 16–PSK are necessary, as in Clover-II. These non-orthogonal modulations require significantly more power to obtain the same bit error rates (BER).

Modern communication theory dictates the opposite approach -- spreading the signal over a significantly higher bandwidth, coupled with the extensive use of FEC (Forward Error Correction) coding. While this may seem counter-intuitive, it is a direct result of Shannon's channel capacity theorem:

$$C = B \log_2(1 + S/N)$$

C, channel capacity, increases linearly with B, bandwidth, but only with the log of the received power (a function of S/N, the signal to noise ratio). Thus, throughput can be increased by exponentially raising power to get more bits per second per hertz (for bandwidth efficiency), or by increasing bandwidth, for more bits per unit energy (for power efficiency).

Theory

Modulation

1 This paper is submitted for use in the Proceedings of the 16th ARRL and TAPR Digital Communications Conference

2 This project was originally started as a class project for EE55 1 in the Spring of 1997. The team consisted of Matthew Ettus, Micheal Batz, and Michael Lu. Continuing development has been by M. Ettus

RTTY, AMTOR, and other common HF modems use binary frequency shift keying (FSK) for data transmission, which is, unfortunately, not very bandwidth or power efficient. Phase shift keying (PSK), and its many variants, have become popular recently in such modes as Clover-II and Pactor. PSK, however, suffers on the HF channel due to such factors as doppler shift, especially at very high data rates.

MFSK is a generaliztion of FSK in which one of m different tones is sent every symbol period, where $m = 2^n$. Each symbol conveys n bits of information. In binary (standard) FSK, $m=2$ and $n=1$. As m increases past 4, the number of tones (m) increases faster than the sumber of bits sent per tone (n), so the bandwidth necessary for a given throughput increases also. The power necessary to maintain a given BER, on the other hand, decreases with increasing m. The symbol error rate is dependent on the energy transmitted for each symbol, but each one carries multiple bits. Thus less energy per bit is necessary. Under moderate $E_b N_0$ ratios, with higher m ($m>=8$), MFSK actually performs better than coherent binary PSK (BPSK).

Another advantage is the ability to perform demodulation noncoherently. This is a very valuable property on ionospheric paths, because strong multipath and doppler make it difficult to recover absolute phase information from received signals.

Forward Error Correction (FEC)

Through the addition of controlled redundancy, Forward Error Correction allows received signals to be decoded successfully, even the presence of limited errors. Thus, less energy per bit need be expended by the transmitter. Lower power transmitters may be used, or higher throughputs can be accomodated.

There are many different FEC methods in use, each with different properties. For this modem, we chose to use Reed-Solomon codes.[3] Reed-Solomon codes are very powerful codes based on Galois field arithmetic.

One property of MFSK is that when an error does occur, it will affect (on average) half of the bits in a given receive symbol. This does not fit well with many other FEC schemes, because they do not handle burst errors well. Because RS codes are based on symbols (groups of bits), rather than individual bits, multiple errors in the same symbol have no more detrimental effect than a single error. This burst error correcting capability also helps on the HF path because many of the sources of noise (i.e. interfering carriers, fading, and burst noise) are bursty in nature.

In our design, we augment the burst error correcting capabilities of the RS codes by interleaving them. Thus consecutive errors are spread across different code blocks, rather than being concentrated in the same one.

System Design

System Specifications

For our modem, we chose to use 16-FSK, in a bandwidth of 2000 Hz (this will fit through the 2.7kHz BW available on most HF SSB radios). Due to the requirement for orthogonality of

3 The Reed-Solomon encoding and decoding software used was adapted from code written by Phil Karn, KA9Q, and others.

symbols, and the noncoherent demodulation which will be discussed below, each tone must be separated in frequency from the next by an amount equal to the symbol rate. To equally space the 16 tones in 2000 hertz, each must be separated by 133.3Hz. 125 was chosen as it is the closest factor of 8000, a convenient sampling rate on PC sound cards. Thus, the symbol rate is 125 baud, and the (raw) bit rate is 4 x 125 = 500 bits per second.

We are using shortened RS codes over $GF(2^8)$ of block length 32 bytes, with 8 of these blocks in a packet. In each block, the modem can be set to use from 2 to 12 redundancy bytes (30 down to 20 data bytes). This allows from 1 to 6 errors to be corrected per block (1 for every 2 redundancy bytes). Each byte is composed of two 4-bit symbols (16 FSK gives 4 bit symbols). The actual data rates are then 3 12 to 469 bits per second.

The 8 blocks are interleaved in a packet, meaning that the first byte of each block is sent, then the second byte of each block, etc. Thus, a burst of errors is spread into separate blcoks. With the highest amount of redundancy, 96 consecutive symbols could be wiped out, and the packet still recovered. A sequence of 8 symbols (the synch vector) is sent as a header for each packet. This allows the receiver to have a known sequence for it to use in synchronization with the transmitted packet.

Modulation

Modulation is performed by generating the appropriate sine wave corresponding to the symbol to be transmitted. This is done by stepping through a table of sine values, and outputting the values to the sound card.

Demodulation

In order to demodulate the incoming signal, synchronization must be acquired first. By correlating the synch vector to the received signal, the modem determines when to start demodulating. The actual demodulation is done by using the fast Fourier transform (FFT). The FFT returns the energy in each frequency band. The band with the highest energy is chosen as the best estimate of the received signal. This is equivalent to having 16 separate filters, and choosing the one with the highest output.

Results

This modem has been tested off of the air, so far, using a baseband simulator running on another PC. Computer to computer file transfers have been accomplished. The next step is on-the-air testing. In testing our modem with a channel simulator, we can see that our results match up well with our theoretical calculations.

The modem performs very well in gaussian noise environments. In order to maintain a 10^{-5} BER, 16-FSK requires 6dB less energy per bit than 2FSK, and 1 dB less than BPSK. With the addition of the Reed-Solomon coding, bttween 3 and 6 dB of additional gain is achieved (depending on the level of redundancy used), giving an overall gain of 9 to 12dB. At the high redundancy rate, 312 bps is achieved with about 12db (1/15) of the power of 300 baud FSK.

We are currently in the process of expanding the channel simulator from simple AWGN (additive white gaussian noise), to include fading and other HF channel effects. The properties of FEC should allow our modem to experience less degradation in these channels than other modems. The FEC can cope with large amounts of incorrect data due to temporary fades.

Implementation

The code for this modem was written in C, for the Linux operating system. It should be portable to any other OS, with minor changes in the sound interface code. The code, when running on a Pentium Pro 150 took less than 5% of the processing time of the computer, so it should function well on much slower computers. A floating point unit is recommended.

Conclusions

We have shown that it is possible to implement a modem entirely on a PC, to save costs, while at the same time, improving on the available options for HF communications. Our design allows higher throughput than the freely available options (AMTOR, etc.) with less than one tenth of the transmitted power. Conversely, one could communicate effectively in conditions more than 10 times worse than is currently possible.

Areas for further study

Continuing work on this modem will expand it in several ways. First, we will further modify the RS encoding and decoding software to allow for »on–the–fly« reconfigurability of the amount of redundancy in each block. Thus, smaller fraction could be used when channel conditions are good, and then increased when conditions get poorer. We also plan on modifying the code to allow the modem to change to higher or lower order MFSK (i.e. 8–, 32–, 64– or even 128–FSK) depending on channel conditions.

Other future work will include integrating the code into a driver so that the Linux system can use it in conjunction with its other networking features, such as TCP/IP and AX.25. This would makc for a complete HF networking system

References

McDermott, Tom. “Wireless Digital Communications: Design and Theory.” Tucson Amateur Packet Radio Corporation, 1996.

Rappaport, Theodore S., “Wireless Communications: Principles & Practice.” Prentice Hall, 1996.

DETECTION & ESTIMATION OF COVERT DS/SS SIGNALS USING HIGHER ORDER STATISTICAL PROCESSING

Mamdouh Gouda, Ernest R Adams,& Peter C J Hill

School of Engineering & Applied Science (SEAS)
Cranfield University RMCS, Shrivenham, Swindon, Wilts, SN6 8LA, UK
Phone: +44(0)1793 785211 Fax: +44(0)1793 783192
E-mail: gouda@rmcs.cranfield.ac.uk

Abstract

Conventional linear and non-linear receivers are generally ineffective in detecting direct-sequence spread spectrum (DS/SS) signals if the spreading sequences are unavailable. An investigation into using correlation-based processing is reported showing that the cyclostationary property of DS/SS provides detection capability. Finally we describe with results an emerging technique based on higher-order statistics where triple correlation analysis is used, leading to the detection and estimation of DS/SS length and its code generating function $g(X)$.

1. Introduction

Modern civil and military electronic communication systems are becoming ever smarter, more complex and in some cases also very difficult to intercept because the nature of the design leads to a covert signal structure. In fact some transmitted waveforms are intentionally designed to make the detection process virtually impossible. Such low-probabilities-of-intercept (LPI) signals have very wideband extremely low power spectral density signatures based on a hidden code structure; moreover they can operate in a complex radio environment of high noise, interference, jamming and other co-channel signals.

A particularly difficult threat signal to intercept is direct-sequence spread spectrum (DS/SS) and in this case the problem is exacerbated when such systems operate in multiple access mode using code division multiple access (CDMA). Moreover, if the pseudo-noise spreading codes are very long and the intercept window is short including an unknown number of aperiodic data modulations then standard signal processing methods based on second order statistics are severely limited.

State of the art techniques for communications signal detection which are based on second-order (i.e. variance) spectral processing are considerably limited by 'phase blindness' and the inability to easily separate out wanted signals from background noise[1]. Our study, however, is focused on higher-order statistical processing which specifically uses the cyclostationary signal property but is combined with the suppression of Gaussianity [2,3] in order to improve the SNR of the detection process. This was the key motivation for our attack on the problem of detecting DS/SS using correlation analysis and also spectral processing for detecting CDMA and chip-code characterisation respectively. The paper discusses the theory of triple correlation function analysis as applied to the detection of DS/SS PN chip code sequences in some detail and then describes the various methods which have been investigated for estimating the basis code polynomials of DS/SS signals in the presence of channel noise. Attention is given to the importance of the *doubling* technique which improves SNR and reduces the dimensional@ of tcf characterization.

2. Higher-order moment signal processing techniques

The proposed detector for covert DS/SS signals uses third order cumulants in the form of triple correlation analysis, bispectral processing and an associated characterisation process[4]. HOM/HOS techniques should be better able to exploit the non-gaussian cyclostationarity of the signal against the channel noise and interference. For example, the shift-and-add property of the m-sequences[5], i.e. $u \oplus u_p = u_q$, where $\oplus$ is binary addition of sequence elements, with an equivalent polynomial representation,

$$g(X) + X^p g(X) \bmod (X^L + 1) = X^q g(X) \bmod (X^L + 1),$$

leads to the delta function response for the periodic autocorrelation function (ACF) $C_{xx}(\tau) = \mathrm{E}[v(t)v(t+\tau)]$ and this ACF is no more than the second order cumulant in HOM terms.

Higher-order cumulants simply extend the averaging process by considering additional time-shifted versions of the same m-sequence signal. In particular, the third-order cumulant or *triple correlation* is defined as

$$C_{xxx}(\tau_1, \tau_2) = \mathrm{E}[v(t)v(t+\tau_1)v(t+\tau_2)]$$

where $\tau_1 = pT_c$ and $\tau_2 = qT_c$ for C_{xxx} sampled at the chip rate $1/T_c$ Hz. In practice, m-sequences use the values ± 1 rather than 0 and 1: $(0,1) \rightarrow (1,-1)$. The previously defined binary addition of sequences, $\oplus$, is equivalent to multiplication, *, in this new domain.

3. Triple correlation of complete m-sequences

The discrete version of $C_{xxx}(\tau_1, \tau_2)$ is evaluated as

$$C(p,q) = \frac{1}{L} \sum_{i=1}^{L} v(i) v_p(i) v_q(i)$$

where $v_j(i) = 1$ if $u_j(i) = 0$ and -1 if $u_j(i) = 1$, i denoting the i-th element of the sequence. By the shift and add property, for certain (p', q'), $u_{p'} \oplus u_{q'} = u$, and it follows that Vi, $v_{p'}(i) v_{q'}(i) = v(i)$. For those pairs:

$$C(p',q') = \frac{1}{L} \sum_{i=1}^{L} [v(i)]^2 = 1.$$

For other (p, q) pairs, $v_p * v_q = v_s$, where $v_s \neq v$ but is an m-sequence by field closure, in which case

$$C(p,q) = \frac{1}{L} \sum_{i=1}^{L} v(i) v_s(i) = -1/L.$$

Thus the shift and add property results in $C(p,q)$ peaks at locations for which $\alpha^{p'} + \alpha^{q'} = 1$. Each peak at (p', q') is mirrored at (q', p'), as $\alpha^{q'} + \alpha^{p'} = 1$. Also, because of the existence and uniqueness of a q'for eachp'in the range $1 \leq p', q' \leq$ L-l, there is exactly one peak in each row and column between 1 and L-1, a total of L-l peaks.

In the original equation for $C(p,q)$, v,(i) = $v(i+p)$ and $v_q(i) = v(i+q)$, i.e. those sequences are advanced in time or shifted to the left (LS). There is a corresponding delayed or right-shifted (RS) version of

C. Substituting $i+q=n$ and assuming $L>q\geq p\geq 0$, $v(i+q)=v(n)$, $v(i+p)=v(n-(q-p))$ *and* $v(i)=v(n-q)$. Thus $C(p,q)$ may be written in terms of RS versions of v:

$$C(p,q) = \frac{1}{L}\sum_{n=1}^{L} v(n)v_{-q}(n)v_{-(q-p)}(n) \qquad \text{[RS]}$$

$$= \frac{1}{L}\sum_{n=1}^{L} v(n)v_{L-q}(n)v_{L-(q-p)}(n) \qquad \text{[LS]}$$

For each peak at (p,q) there is a corresponding peak at $(L-q, L-(q-p))$; reflections are also peaks. E.g. as sequence *a* ,with generating polynomial $g(X)=X^5+X^2+1$, (L=31) has peak $(p,q)=(1,18)$, it also has peak (13,14); (18,1) and (14,13) are also peaks as shown in Fig 1.

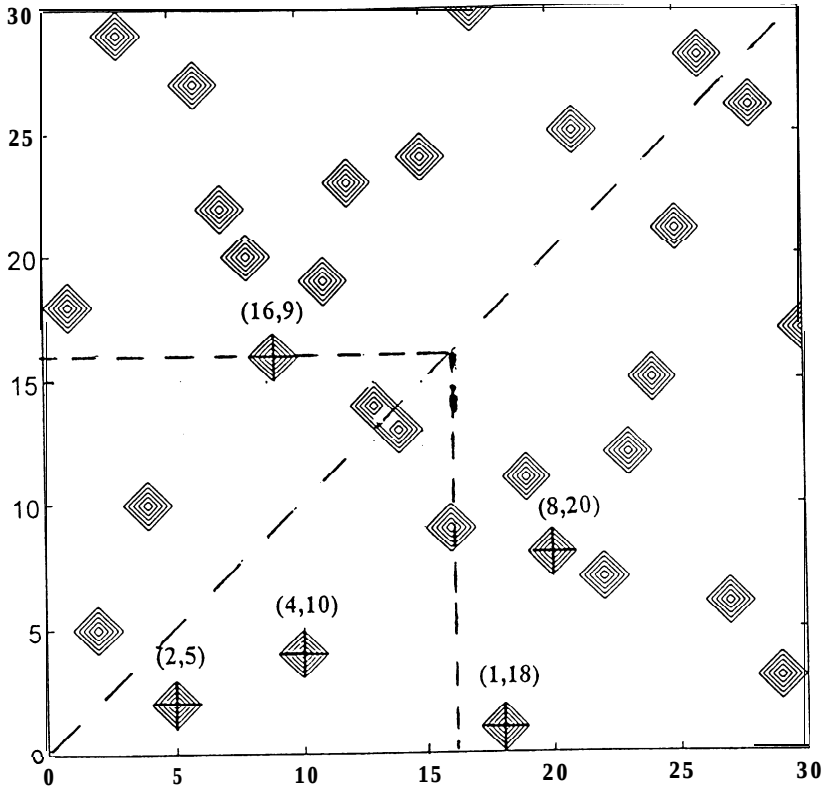


Fig 1. Coset summing for partial tcf [45]

4. Determination of *g(X)* from triple correlation peak locations

The theory developed so far assumes the m-sequence length L is known. In particular, knowledge of L is necessary to evaluate the triple correlation C(p, q). However, there is evidence that sufficiently long partial m-sequences produce good estimation of peak locations. L may be derived from the peak locations

$$(i, j), (2i, 2j), \ldots, (2^k i, r) \quad \text{for } i < j$$

Assuming the last pair is the first for which $r < 2^k j$, $r = 2^k j \bmod(L)$, or $L = 2^k j - r$. As an example, the pairs (1, 18),(2, 5) would produce L = 36 - 5 = 31. As peaks may be displaced, several such sequences should be examined.

If L is known, it is possible to determine $g(X)$, and thus the tap weights of its LFSR, from a *single* triple correlation peak location. This follows from the fact that, for a given L, different primitive $g(X)$ produce no common peaks.

As an illustration, consider the m-sequences of length 31 generated by $g(X) = X^5 + X^2 + 1$ (45 octal). If, for example,as shown in Fig 1, the peak location (1, 18) is known, the following peak locations may be predicted:

$$(1, 18), (2, 5), (4, 10), (8, 20), (16, 9).$$

From the one peak such as (2, 5) we can derive $g(X)$:

$$\left.\begin{aligned}\alpha^2 + \alpha^5 = 1 \Rightarrow \alpha^5 = \alpha^2 + 1\end{aligned}\right\} \Rightarrow \alpha^5 + \alpha^2 + 1 = 0.$$

Thus a is a solution of $g(X) = 0$, where $g(X)$ must be of order 5 and include 1:

$$g(X) = X^5 + X^2 + 1.$$

5. Coset summing for better detection

Although the effects of noise may be reduced by averaging the tcfs from several short signal samples, this process is computationally costly and assumes persisting m-sequences. Coset summing is an alternative or complementary technique which improves SNR and reduces the dimensionality of tcf characterization. It may lead either to powerful multivariate discrimination of fragments of known m-sequences or to the blind determination of an m-sequence from the detection of a single tcf peak.

Coset summing involves searching NxN partial tcfs for peaks by using the *doubling* property of peak locations. Each feasible tcf location is visited, beginning $(p,q)=(1,2),(1,3),...,(2,3),(2,4),...$, and the following sum of non-repeated tcf values calculated:

$$S = C'(p,q) + C'(2p \bmod L, 2q \bmod L) + ... + C'(2^r p \bmod L, 2^r q \bmod L)$$

where $p=2^{r+1}p \bmod L$ and $q=2^{r+1}q \bmod L$. Each doubled location, $(2^i p \bmod L, 2^i q \bmod L)$ for $1 \le i \le r$, is excluded from the future search as its coset is represented by the initial location (p,q). Thus each coset present in the partial tcf is represented by a single peak, called the coset leader. Clearly, all the peaks of a coset will not generally be present in the partial tcf. When doubled locations lie outside the partial tcf ($p', q' > N$) no contribution is made to the sum but doubling is continued, and further values for locations within the partial window summed, until the original (p,q) results.

Coset summing is illustrated in Fig 1, the tcf of a 3 1 length m-sequence [45]. Assuming only a 16x16 partial window is available, the peak at (2,5) would be the first peak encountered in the search. The other coset peaks added would be at (4,10) and (16,9): doubling would generate the other coset members (8,20) & (1 ,18) lying outside the partial window. However, the peaks at (4,10) & (16,9) would be excluded from the search and their contributions included at (2,5), the coset leader.

Clearly, in addition to reducing dimensionality, the use of coset sums of available tcf values will improve the detectability of actual peaks. Actual peaks of partial tcfs of m-sequences in noise have average values of 1 while the average non-peak values are $-1/L$. If the search begins with a non-peak location, other locations generated by doubling will also be non-peak.. Thus all the coset sums will consist of exclusively peak values or exclusively non-peak values. These summed values may be tested against a threshold to decide whether they arise from a coset of actual peaks.

6. Conclusions

Triple correlation analysis provides a powerful means of searching for and detecting the presence of covert wideband signals such as DS/SS. The results show that the tcf is an excellent basis for detection and identification of m-sequences. The doubling process (coset sum)improves the detectability of actual triple correlation peak and reduce non-peak values: However higher-order statistical processing can extract more information than that conveyed by second-order power spectral density or autocorrelation function.

7. References

1. Stephens J.P., 'Advances in signal processing technology for electronic warfare', Journal of Electronic Defense, September 1995.

2. Gardner W.A., 'Signal interception: a unifying theoretical framework for feature detection', IEEE Trans on Comms, 36, 1988.

3. Nikias C. L. & Petropulu A. P., 'Higher-order spectra analysis - a nonlinear signal processing framework', PTR Prentice-Hall Inc., 1993.

4. E R Adams, M Gouda & P C J Hill, 'Detection and characterisation of DS/SS signals using higher-order correlation', Proceedings of IEEE 4th ISSSTA, Mainz, Germany,Sep 1996, pp27-3 1.

5. S. W.Golomb, Shift Register Sequences, Rev.Ed.(Aegean Park Press, 1982).

HamWeb: Rethinking Packet Radio

by John Hansen, WAOPTV
State University of New York at Fredonia

Abstract: This paper describes a general purpose implementation of a simple "broadcast protocol" useful for terrestrial amateur packet links. It allows the transfer of files and entire directory structures from a server to many client stations simultaneously. Consideration is also give to applications of HTML to amateur packet links.

Keywords: Packet, Broadcast Protocol, HTML

Once upon a time I thought there was only one issue with regard to speed and packet radio: how to get more of it. About six years ago I spent a considerable amount of time and effort arguing in favor of abandoning 1200 baud packet in favor of 9600 baud, and ultimately higher speeds. Back then I was running a packet radio BBS and went so far as to open a user port that ran at 9600 baud. It saw extremely little use. I was thinking of offering certificates for the first 5 users of that port, but never got that many takers.

At first, it seemed like the barrier to higher speeds was the lack of commercially-available equipment. While you could buy an add-on modem for your TNC for about $100, installing it was no cakewalk. Furthermore, at that time no commercially-available transceiver would support 9600 baud out of the box. Finally the packet equipment manufacturers started to offer TNC's with 9600 baud built in and some radio manufactures began to offer 9600 baud-ready radios. However, there were still very few takers.

Back when packet radio was getting started, the most commonly used telephone modem ran at 300 baud. TNC's capable of 1200 baud must have seemed fast. I know I was impressed by the speed when I first got on packet. Now the most common telephone modems are 28,800 baud and the most common packet modems are (you guessed it) still 1200 baud. And, according to recent reports, conventional packet radio is withering.

Why did higher speed packet never catch on for end users? The conventional wisdom is that ham radio operators are both too cheap and too lazy to take on the financial and technical challenges of higher speed digital communication. You know the argument... ham radio operators used to be technically oriented, home constructors, but now they are just appliance operators. An ancillary argument is that this is somehow all related to the no code license. This bit of folklore has been repeated so often that almost everyone has begun to believe it. Like most conventional wisdom, it is in large part incorrect.

A couple of years ago at the Rochester, NY Hamfest I had a discussion with Steve Ford, WB8IMY of the ARRL. He raised an interesting point, namely, what would we do with the speed if we got it? Suppose, hypothetically, that overnight every 1200 baud packet TNC/radio

combination in the country was magically converted into a 9600 baud station (or a 19,200 baud station). How would that change anything? You would still have a systern based largely on text BBS's handling roughly 500 messages per day, where most users read and compose messages online. Few people can read faster than a solid 1200 baud packet connection provides data, thus for these applications they see little point in pulling in data any faster. As long as the basic data infrastructure remains the same there is no incentive for speed.

So we have an interesting conundrum: End users don't want to invest in higher speed packet unless there is something interesting and fundamentally different that they can do with it. On the other hand, interesting new applications that might utilize higher speed packet are not being developed because of the lack of potential users. As a result many former users of packet have packed up their TNC's and put them away because they find the Internet more interesting.

Is this really inevitable? For a few months there I thought so. Now I'm not so sure. Any solution to the current packet problem must present a clear migration path that will provide interesting new applications for 1200 baud packet users that will both capture their imagination and provide some real justification for the expense and trouble of moving to higher speeds. That is, this new system must not be predicated on the assumption that we will all operate at faster speeds, rather, it must work reasonably well at 1200 baud. But it is equally important that it offer something genuinely different than what users are seeing on the current crop of packet BBS's

That's a tall order all right, but fortunately many of the building blocks for such a system are already within our grasp. First, we must build on our strengths. From a technological standpoint, what advantages does radio offer over wires (or fiber... the technology that lies behind the Internet)? I think that there are two clear advantages to radio: mobile operation is much easier with radio and radio is inherently a point to multi-point technology. For our purposes here it is the latter item that is significant. Data communication that occurs at slower speed can actually be more efficient than higher speed communication if the slower speed method also transmits data to many users all at once. If you need to communicate, say 100K of information to 100 different people, 1200 baud is actually faster than a 28,800 baud modem (and even faster than 56,000 bps frame relay technology) if the 1200 baud signal can reach everyone at once, but the higher speed signal has to retransmit the information separately to each of the 100 destination stations.

Imagine how little would be accomplished by a AM broadcast news station if it had to read the news to each listener individually rather than broadcasting it to everyone at once. Imagine how poorly our amateur radio voice nets would run if the net control station had to repeat each transmission over and over again, once to each station that checked in. Well, folks, this is the way we do packet radio. If 100 of us are interested in reading the same message we each connect to the BBS and download the message. None of us hear the other's download, so we must each ask for the transmission separately.

With the exception of a few of the newer technologies, the Internet works exactly the same way. When you connect to a chat server, or a CUSEEME server, the server receives what each connected station sends and then retransmits it individually to each of the stations involved in the conversation. If there are twenty participants in the chat, the server must repeat the information twenty times.

With radio there is no inherent reason that we have to be so inefficient. In fact, not all of packet radio is this inefficient. The digital satellites, for example, have for years been relying on an orbiting server that broadcasts packet frames. They can be received at many, many stations all at the same time. If one or more of the stations misses something, it can make a request for the server to fill the hole, rather than retransmit the whole file. Believe it or not, this is not the way the system was originally envisioned. The first orbiting packet station was aboard the Japanese satellite JAW. It worked pretty much like our terrestrial BBS's work, only with far fewer commands. The original plan for the microsats was similar... it was to orbit a WORLI-compatible BBS (which was then the world standard). I was at the annual AMSAT meeting in Des Moines when Harold Price (NK6K), one of the principle software designers for the spacecraft, announced that the WORLI-compatible system wouldn't work. He did some quick calculations about the number of potential users and the fact the satellites would only be visible for a maximum of about 15 minutes per pass and concluded that if everyone was to try to connect to the satellite individually, very little data would actually get through. This gave rise to the concept of broadcasting files for everyone to receive at the same time. This is essentially the system that is still in use today.

For the 1994 AMSAT meeting in Houston, I presented a paper that argued that this same technology could be put into practice on terrestrial BBS systems. I included some code I had written to construct AX.25 frames and broadcast them through a G8BPQ switch (which was the technique that was commonly used by BBS's at the time). At the end of the article I invited anyone who wanted to, to steal the idea and make it into a workable system.

Over the past year my thoughts have been returning to the proposal I made in 94 and the comments that Steve Ford made last summer. Sometimes it takes a while for all the pieces of an idea to come together, but I think they finally have. What we need is both a means of transmitting far more data to users in a shorter period of time, and a rethinking of the content of those transmissions.

So starting first with the underlying technology, it is necessary to be able to communicate a lot more data. This must be done, however, without requiring the end users to make a substantial investment in new equipment. I think sending frames in an unconnected mode (as occurs with the satellites) is the clear answer here. Not only does the server communicate with more than one station at time, but a lot of time is also saved by getting rid of the acknowledgement packets.

Currently a packet station sends out a transmission of roughly 1000 bytes and then waits for the other station to acknowledge receipt before sending more. Those who have done the calculations say that this causes the 1200 baud data rate to achieve a throughput rate under the most optimal conditions, with no nodes or digipeters, of about 600 bits/sec. In the real world the throughput is most often substantially less. After taking into account overhead, 600 bits/sec will give you about 60 bytes (or characters) per second. That's about 600 words per minute, which is a pretty good reading speed, but not great for file transfers.

The server that I've written (dubbed "HamWeb", for reasons that will become clear shortly) has achieved measured throughput rates of over 115 bytes (or characters) per second. That is, the speed is about twice that of the conventional packet channel when the conventional channel is running at its fastest. And this is using the same basic 1200 baud equipment that most packeteers already own. But this is not where the real throughput boost occurs. It is even more important that the HamWeb Server allows these files to be received by many stations at the same time. The more receivers that are turned on at once, the more efficient the system becomes.

The basic technology building block, then, is a server that pumps data continuously out a serial port and into a TNC. No receiver needs to be hooked to the TNC, because if the server is going to receive anything, it will be on some other channel (with another radio and TNC) instead of the one where the continuous transmitting occurs. However, the transmitter to be used in this application had better be able to withstand a very high duty cycle, because it will essentially be always on. It is possible that commercial transmit strips my be useful for this purpose. Obviously, since the receiver is being omitted, the T/R switching can be dispensed with as well. The server software takes the files that have been specified (or entire directory structures), breaks them down in to smaller parts, adds some additional information concerning the filename, size, etc. as well as a "checksum" to ensure that the data is received error-free, and sends the data to the TNC.

On the user side, the minimum required station is an FM receiver and a TNC that is capable of running in "KISS" mode (virtually all of them are). What's more, it is possible (though I've not done it) to write an end user program that would use the station computer's sound card instead of a TNC. In this instance, one could simply buy a $50 receiver and run the audio output into the mic input of the sound card. If the information available through this system is sufficiently interesting, I think it is reasonable to think that a significant number of non-ham hobbyists might set up receiving systems to monitor these transmissions in the same way that they currently use scanners to monitor repeater traffic.

The software running on the client computer generally will run in the background and take the incoming packets and reconstruct the files so they appear just as they did on the server. If part of a file is missing, the client keeps track of the pieces that it needs and grabs them the next time they are transmitted, until a complete file is constructed. I assume that the typical user will simply leave his receiver on all the time and collect data around the clock.

So that's the basic file sending/receiving engine. But as Steve Ford said, if all we've done is radically speed up the transmission of data without changing the nature of the service . . . well, what's the point? For example, those who operate the large BBS's in our area tell me that they currently receive about 500 bulletins a day. Assuming an average bulletin is 1000 characters in length (many are longer, but most are shorter), the system described above could transfer every one of those bulletins to every receiving computer in its area in under one hour. I don't mean a listing of the titles of the messages (such as is done by TPK or WinPacket) but rather the entire contents of all of the bulletins that an average BBS receives in 24 hours. Even if we wish to stick with the contents currently available on packet radio, we'll have to think of something for these servers to do the other 23 hours of the day!

Communication on the Internet is based on a language called "Hyper-Text Markup Language" or HTML. Those who have designed their own web pages will be intimately familiar with this language. Once upon a time it was relatively difficult to do this because HTML itself is rather arcane. However, many new software packages (some available for free) have become available that allow folks with absolutely no background in computer programming to design very attractive and functional web pages.

When I first began to explore HTML the thing that most impressed me was its efficiency. Beautiful pages that appeared to have bit-mapped graphics could be put together in only a few kilobytes of data. This is possible because the formatting of the data is actually done on the end user machine, not on the web server. Try the following experiment to see what I mean. Go to one of your favorite web pages with Netscape. Click on "File" and then click on "Save As". Give your file and name and save it. Now switch from Netscape to a listing of the directory where your file is saved and examine the size of that file. I think you will be surprised at how small it is. Graphics, of course, require additional space, but most web page graphics are highly compressed versions of rather small files.

When I first discovered this, what came to mind immediately was that HTML seemed ideally suited to amateur radio applications. It seemed to me to be obvious that the next generation of BBS software would use HTML or something like it to provide information in a much more attractive and useful way. However, I didn't pursue this because I had neither the time nor the expertise to design a new BBS system around HTML. Furthermore, suppose you could design such a BBS. Given the huge installed base of BBS's using the forwarding scheme that traces its routes back to W0RLI's original code, unless this new, improved BBS system was completely compatible with that forwarding system, it was probably doomed to obscurity.

I believe that there is now a possible transition path between older BBS's and new systems that can provide a wide array of services. The key is to use Netscape (or some other browser) as the principal terminal program for utilizing the packet radio store and forward system. The kids down in Mountain View and up in Redmond have spent a lot of time and effort creating these web browsers. We should simply use their technology building blocks for another purpose. This

is possible because "Web Browsers" are not simply programs for browsing the world wide web. They are more general purpose programs which, among other things, can be used for viewing any kind of HTML document.

What I am suggesting is that we use a HamWeb broadcast server to provide local distribution of amateur-radio related information in the form of world wide web pages. End users would run the HamWeb client software in the background to continuously receive these files and dump them into a directory on their hard drives. The HTML documents would then be viewed using any standard WWW browser. Every morning you could wake up and have a completely new electronic amateur radio magazine delivered to your computer. It would have the type of attractive formatting and color that we are used to seeing in web pages. It could have graphics, pictures, even sound. What's more, it could also contain links to world wide web sites, so that if you happened to be simultaneously hooked into the web, you could move seamlessly between information that was provided by radio and information that was available on the web. And in addition, the information that came in by radio would appear to the end user to show up on the screen almost instantaneously, since it would be read directly off his hard drive.

As an isolated system this would provide a very interesting experiment. However, no local community of amateurs is going to develop enough content to provide interesting reading on a regular basis. Instead, we must find a way to layer this system on top of the existing packet forwarding system. If the pool of potential producers of web pages to be distributed in this fashion is extended to the entire globe, then we would begin to see enough innovative content to make such a system worthwhile. Standards would need to be worked out for identifying and forwarding HamWeb pages to insure that file names are not duplicated. I think the way to do this would be to have each set of web pages forwarded as a unit, an encapsulated compressed file. The filename would consist of the home or index page for the set with the originating BBS's callsign tacked on (and perhaps a serial number). Then as the set is received by each server in the network it would be placed into its own subdirectory. Since the HamWeb client/server software causes the entire subdirectory structure to be duplicated on each of the client machines, there should be no problem with individuals writing pages with identical names.

The software that I've written provides a general purpose mechanism for moving data from a central server to a lot of client computers, all at the same time. I have tried to write this program as generally as possible to lend itself to a wide variety of applications beyond serving up web pages. A number of "hooks" have been built in to allow the server to interface with other programs to initiate file broadcasts.

Another interesting application, for example, would be to use the server on a satellite ground station to replicate all of the directory files and message files from the PACSATS on computers in a given region. Thus, anyone who wanted to could run any of the available PACSAT groundstation programs without actually putting up a satellite station themselves.

In emergencies, a HamWeb server would provide a mechanism to distribute large quantities of information (including graphics and other multimedia files) over a wide region. Most importantly, the receiving points would not even have to be equipped with transmitters. Scanners would be adequate to provide the latest updates of information.

The HamWeb Server and Client software are now available in beta version on the TAPR host at:

www.tapr.org/~wa0ptv

or by ftp at:

ftp.tapr.org in the pub/wa0ptv directory

Appendix: Technical Documentation for the HamWeb Server

The HamWeb server is a program designed to broadcast information using amateur packet radio. I have written a client program that collects these broadcast files which runs under Windows 95. However, others may wish to collect the data for other platforms or may wish to construct improved client programs for Windows 95. As a result, this appendix will provide the specifications of the packets that are transmitted by the HamWeb server.

The HamWeb server communicates with the TNC using the KISS protocol. I am assuming that this protocol will also be used on the client side. The KISS protocol starts each packet with a CO and ends each with a CO. To ensure that CO does not occur anywhere else in the packet (thereby causing the client program to think the packet is over) it is necessary to make some substitutions as the data is transmitted. The client program is therefore responsible for undoing these substitutions on the receiving end. For more complete documentation of this process see the Kiss docs on the TAPR website by Mike Chepponis and Phil Karn. In constructing a HamWeb client, your responsibilities with regard to undoing the substitutions will be discharged if you scan through the packet for the byte DB. In any instance where you find the DB byte, if the next byte is a DC, you should substitute the value CO. If the byte that follows the DB is a DD, you should leave the DB and eliminate the DD. Note in both these cases you are substituting a single byte where there were previously two bytes, so the overall length of the packet will decrease. After the substitutions are completed, you will have the raw data from which you may proceed.

THE HEADER

The corrected data from the TNC may usefully be thought of as being in two parts: An AX.25 header and the actual data. The first byte, of course, is a CO. This is followed by a 00 byte. The 00 byte indicates that the packet contains data rather than a command to the TNC. The next 16 bytes of the packet contain information on the source and destination callsigns and ssid's. Generally they can be discarded. If you are interested in the formatting of this information see the AX.25 protocol document on the TAPR web pages or my paper in the 1993 Proceedings of the AMSAT-NA Space Symposium. From the standpoint of receiving broadcast files, unless you are trying to keep track of the callsign of the server, you may throw away this entire 18 byte header.

THE DATA

That leaves the data. For reference purposes I will call the first byte of the data portion byte 0 even though it is not the first byte received from the TNC. I'm assuming here that you have either discarded the header information or extracted what you needed from it and then discarded it. The first four bytes of the data portion (bytes O-3) are a checksum. The checksum is constructed by adding the actual values of the rest of the bytes in the data portion of the packet starting with byte 4. This will allow you to test to see if the data that you are receiving has become corrupted The

checksum (as well as all other numerical values) are expressed in the standard format of least significant byte first. Thus, the four byte series:

12 AB 28 00

would translate to an decimal value of:

2,665,234

Bytes 4 and 5 are a serial number. Each file is transmitted 1024 characters at a time. The server breaks the file up into 1024 character sections and gives each one a serial number. The first section is serial number 00, the second 01, etc. I have used serial number rather than the actual file offset to reduce the overhead of the system. A two byte serial number will allow single files as large as 64 megabytes to be transmitted. To accomplish this using offsets instead, I would have to use a 4 byte value. The next four bytes (bytes 6-9) are the file size. The following four bytes (bytes 10-13) are the time the file was last modified. This is recorded in the standard format of the number of seconds since Jan 1, 1970. The time is included in the broadcast so that the client software can check whether an existing copy of this file is older than the one currently being received. Thus files can be updated only when they are newer than the existing version. One use for this would be if the server maintained an "index" page of the files available. Thus when the sysop modifies the index page to add or delete files, this page would automatically be modified on all the client computers as well.

Byte 14 starts the path/filename. When the server is replicating directories rather than just transmitting files, the entire path is transmitted in this field rather than just the filename. In writing client software, you are responsible for parsing this path/filename and creating the directory for the file if it does not already exist. I arbitrarily limited the path/filename to 128 characters. You will know that the server is transmitting files to be placed in directories because the path/filename will start with a "\".

Even when the server is simply transmitting files, you should not assume that filenames will be in an 8.3 format. Web page filenames are commonly much longer than this. Longer filenames are supported by Win95, WinNT, Mac system 7.5, and virtually all flavors of UNIX. Thus it seems silly not to use this capability. As a result however, you do not know how long the filename will be before you receive it. I have added an FF character to mark the end of the filename. All characters starting with byte 14 and ending with the character before the FF constitute the filename.

All characters after the FF are the actual data contained in the file. For the most part, this will be 1024 characters, since the file is broken into those chunks. However, you can not assume that the data will be exactly 1024 characters long, since the last chunk of any given file is likely to be less than 1024 characters. The packet ends with a character CO, which also may be discarded. Note

that you must identify where the packet ends before you perform the DB/DC to CO conversion or you may be fooled into thinking the packet ends before it really does.

OBSERVATIONS AND HINTS

Some may wonder why I didn't simply use the PACSAT protocol that is currently used on amateur satellites. There are a number of reasons. First it contains a whole lot of information that is simply never used for anything. This is not a criticism of the folks who wrote this protocol. At the time it was written we had absolutely no operational experience with it and it probably seemed like many items (such as the number of times a file had been downloaded) were useful. The PACSAT protocol creators were also writing a system that could be implemented on DOS computers (the most popular operating system for PC's of that day) and so shorter filenames (which in the case of the PACSATS were numbers) were essential. Because I expect that most people will view these files with a web browser rather than a terminal program, my only goal in this project is to transfer files from the server to the clients and provide enough information to manage the files on both systems. Thus, I've opted for a substantially simpler protocol.

In writing your client software you should assume that any character could be lost in transmission. Thus, any time you are filling a array by waiting for an ending character (such as waiting for a CO to know the packet has ended, or waiting for an FF to know that the filename is ended) make sure that if that character is lost something will stop the array from exceeding the limit you set for it. Otherwise the results could be very nasty.

Finally, make certain that you keep careful track of the length of the packet. Every time you replace a DB/DC with a CO or a DB/DD with a DB you must make sure you reduce the length of the packet by 1.

Wireless In Ulaan Bataar

Learning From A True-Life Mongolian Network Adventure

BY

Dewayne Hendricks, WA8DZP

In Ulaan Baatar, Mongolia, severe weather conditions prevail, the wired telecommunications infrastructure is very poor, advanced telecommunications technology expertise is limited (although there is considerable local computer expertise), and US access to Mongolian scientific and research facilities is highly constrained by lack of normal Internet connections.

Last year, some of us went to Mongolia to integrate a series of data radios into a wireless network, and then field-test them. Our purpose was to build on and apply knowledge being gained from the "Wireless Field Test (WFT) Project for Education," funded by the National Science Foundation (NSF) and run by Dave Hughes of Old Colorado City Communications, in Colorado Springs, CO.

The Problem

The expedition's mission was an attempt to remedy the lack of a suitable wired telecommunications infrastructure in Mongolia, including across its capital city--a problem typical of thousands of cities in underdeveloped countries. "Suitable," in this case, means public or private circuits capable of carrying data with the reliability and bandwidths necessary for scientific research between institutions, increasingly equipped with advanced in-house computer and even local area networks.

As a result of this deficiency, the primary institutions in Mongolia, such as the Mongolian Technical University, the Mongolian Academy of Sciences and the many other educational facilities which reside in the capital city are inaccessible (via data link) to US scientific institutions, researchers and students.

What exists is a poor and obsolete government postal, telephone, and telegraph voice-telephone system, a holdover from when the Soviet Union controlled Mongolia. Efforts to use even late model error-correcting phone datamodems to link the computers in the various institutions is only partially successful, since the non-digital circuits are so old. It is difficult to maintain even a 14.4Kbps connection without the line dropping many times in an hour.

As a result of the first step to provide Internet access to Mongolia, NSF funded the establishment of a costly ($72K/yr) satellite Internet link. This 128K link, provided by Sprintlink, only reached one set of computers in a single building, in the central part of the city, by

direct wiring. The PTT has neither the resources nor the expertise to provide even the lowest speed (56Kbps) wired circuits to connect the numerous other institutions.

The "last mile" problem of connecting this satellite link to the institutions (actually about one to ten kilometers from the satellite ground station), which must then distribute the signal to their own wired intranets, had so far defeated efforts to link it by local telephone facilities. In this respect, Mongolia is typical of scores of underdeveloped nations with a poor telephone infrastructure, too limited to adapt to modern requirements for institutional datalinks to and from other nations, especially the US. Wireless datalinks provide one possible general solution to this problem, not only in Mongolia, but in many other countries.

Challenges

A number of challenges made this a difficult project, as well as one from which valuable lessons and techniques were learned for application elsewhere:

* The weather is more severe in Ulaan Baatar than in the areas in Southern Colorado where our wireless field-test project has been implemented. High winds and average temperatures from -15° to -25°C for entire winter months will subject the radios, connectors, shielding materials, and antennas to extremes of cold and wind.

The question was whether or not economical commercial-grade radios and systems could handle the weather and give similar MTBF (mean time between failure) times to those systems operating in milder climates.

* Whether the electromagnetic environment from the equipment that exists in Mongolia today, some from systems developed and deployed by the Soviet Union in the past, in a different technological regulatory environment, will affect the spread-spectrum dataradios that we intended to deploy, which were developed in the US and operate in environments controlled by FCC rules.

* Could the wireless network be integrated into networks made up of a mish-mash of US and foreign computer and peripheral equipment, and would interoperability be a problem?

* Given the level of in-country expertise able to help install and maintain the wireless network, could local personnel be trained sufficiently in a short period of time to handle the network in the future, with only the assistance of remote expertise to call upon for help?

Groundwork

The Mongolian government had permitted a group of Russian-trained Mongolian engineers to form a private technology company, DataCom Co., Ltd, headed by Dr. Dangaasuren Enkhbat, using facilities at a technical center once occupied by Soviet engineers.

The NSF had turned to this company--with limited liquid assets, but in a suitable central facility in the capital--to provide the critical organizational link between the satellite-based Internet feed and in-country institutions. In return for being able to eventually provide Internet

service to private companies in Mongolia, and being given control of the satellite groundstation, DataCom agreed to provide links to Mongolian public, scientific, educational, government, and library institutions for at least two years, at no cost. With initial finding from various sources, and with help from groups such as Sprint, PanAmSat, Comstream, NSF and the us ambassador to Mongolia at the time, Donald C. Johnson, DataCom became the first Internet Service Provider in Mongolia in December of 1995. (The top-level domain name for Mongolia is MN. DataCom maintains a Web site at http://www.magicnet.mn/).

Once the project was approved by NSF, our first task was to determine what would be required for the expedition and what we'd have to do once we arrived in Ulaan Baatar to deploy the wireless network. We made use of maps, videos and communications with the staff of DataCom, in order to determine exactly what types of radios would work in their environment, and what configurations and physical distributions of sites would be required for distribution of Internet services from the DataCom location. The plan was to install a basic wireless Metropolitan Area Network that would consist of eight sites: the DataCom site, and seven sites selected by DataCom. These were all universities, with the exception of the US Embassy.

Based upon our experience in the WFT project, we had made a preliminary decision to use unlicensed Part 15 dataradios developed and marketed in the US by a small Boulder, CO company called FreeWave Technologies, Inc. (http://www.freewave.com/). They make a spread-spectrum radio that can send data at 170Kbps over the air and interfaces to a computer via a serial data interface (RS-232) at 115.2 Kbps. The radio operates in the 902-928MHz part of the spectrum, one of three bands utilized by Part 15 devices in the US. These radios put out 1 watt of power; using various a.ıtenna configurations we have been able to obtain/send data with them distances of up to 60 miles. (For more information on the use of these sorts of unlicensed spread-spectrum radios, check out our WFT project Web site at http://wireless.oldcolo.com/).

We learned that the 902-928MHz band was not completely available for our use in Mongolia, since the commercial cellular telephone network there used GSM technology which operated in a portion of the band from 902-915 MHz. In order to use the FreeWave radios in Mongolia, we had to make two things happen.

We first got DataCom to obtain from the Mongolian government permission to operate the radios in the 915-928MHz part of the spectrum, avoiding the conflict with the GSM cellular network. We next got FreeWave to provide us with a special version of their radio which would operate in this smaller portion of the 902 MHz band. As it turns out, this was not difficult--they had already encountered the same problem when they first entered the Australian market, which uses GSM cellular in the same band.

In order to connect each site's radio to the local LAN we determined that we needed a low-cost Internet router that could be installed and maintained at each location. For the WFT project, we had already developed such a router based upon the widely available Linux operating system and a low-cost PC. We

decided to use this approach in Mongolia, as we already had a great deal of experience with this router and an attached FreeWave radio. Aside from Linux, all a PC needed to function as a special purpose IP router was a high-speed serial card that would support an interface speed of 115.2Kbps and enough RAM to allow Linux to function properly (in this case we used 16MB).

Also important were the antennas and feedlines. It was clear that we needed an omni-directional antenna for the DataCom site, since it was the point-of-presence (POP) for all Internet services and each remote site would have to connect to it. For the remote sites, we would require a directional antenna that could focus all of the available energy from the radio on the POP.

Although the distances involved in the Ulaan Baatar network would be a good deal less than we had been used to in the US (average about 20 miles), we decided to use the same high-gain omni and directional antennas that had used here rather then taking the time to find lower-gain alternatives--it's better to have more antenna gain than you need than to get there with less and find out that you need more. We also decided to use the same low-loss coax feedline (LMR-400) to connect the radio to the antenna that we had been using in the WFT project.

Even with all the up-front analysis of each site's requirements, we still had no idea of the exact amount of feedline that we would require. As a result, we decided to take three times as much as we thought we'd need.

Finally, there was the issue of electrical power. In Mongolia, they use 220 VAC; we had to obtain the necessary power adapters and connectors to insure that the radios and routers could all operate on local AC power.

As it's common practice with an expedition of this sort to a remote part of the world, we made sure that we had spares for all of the key equipment, especially routers and radios. When it came to just how we were going to get all of the equipment to Mongolia, the NSF intervened and made arrangements with the US ambassador in Mongolia that allowed us to ship everything via diplomatic channels (the famous "diplomatic pouch"). Thus we were able to avoid the normal problems encountered using commercial shipping firms and dealing with customs procedures.

It turns out that spread-spectrum radios cannot be exported to countries such as Mongolia without a special export permit from the US government. Obtaining such a permit can take anywhere from three to six months. As we were trying to get the expedition completed before the harsh Mongolian winter started (December-March) we were fortunate in being able to bypass that requirement by shipping the radios the way that we did.

The downside in using the diplomatic channel for shipping is that once you put your shipment into the system, you only have a rough idea when it is going to arrive at its destination. Thus we had to use estimates on the time it took other items to arrive in Mongolia as the basis for determining just when the expedition team itself should arrive. As it turned out, we were lucky in that most of the equipment arrived before we arrived in Mongolia, with the exception of a few pieces that arrived after we had been in-country about a week.

Installation

A team consisting of Glenn Tenney, AA6ER (of Fantasia Systems, San Mateo) and myself left for Mongolia in October of last year. After we had arrived, made the initial visit to DataCom, and inventoried our equipment that had arrived (except for the few missing items), the first order of business was to survey all of the chosen sites in order to determine just what would be needed to deploy the equipment.

We were shocked to find that DataCom was surrounded by high-rise buildings that effectively blocked line-of-sight to all of the sites where installations were to have taken place. One of the key factors in getting this type of low-power radio to work properly is to have a clear line-of-sight path to any location that you wish to contact. Without major obstructions, it is possible to establish communications over fairly long distances; with obstructions it is usually not possible to go more than a few hundred feet. Given what we saw from the DataCom site, it looked like it was time to wrap things up and get on the way back home. However, we decided to make the best of a bad situation, attempt to proceed anyway, and see just what could be accomplished.

First, we installed the first radio and router at the DataCom site. In order to maximize the power output to the antenna from the radio, we mounted the radio on the roof of the building just a few feet from the antenna in a small equipment shed. This approach allowed us to use just a short coax feedline to the antenna and then a rather long serial-data cable to the router which was on the second floor of a four-story building. Since this location was the main hub for the wireless network., we used an omnidirectional antenna.

We next set up a mobile radio so that it could be powered by batteries and carried about in a car with one of the directional antennas. We decided that we wanted to travel around town and see if it was possible to receive the signal at any distance from the POP location. To our surprise, we were able to get a good signal at most of the sites where we were to have installed a radio--all without having line-of-sight to the POP. After some analysis, we were able to determine that even though Ulaan Baatar is populated with a large number of high-rise buildings, the materials used in their construction are such that they appear to be effectively transparent at the frequency of the radio emissions that we were using. This windfall allowed us to proceed with installation at each site as planned.

The most difficult part of the installation is the proper placement of the antenna and its connection to the radio. The major goal is to minimize the length of feedline used to connect the radio to the antenna--the basic rule being that the longer the feedline, the less power actually gets to the antenna, and hence the lower the signal quality at the remote location. This is where we spent most of our time during the next two and a half weeks. Each site had to be surveyed; then we had to work with the local people in charge of that site to convey to them what needed to be done and make sure that the proper permissions obtained to do the work.

At some sites, nothing major was required, as the antenna and the radio/router ended up being in the same room. In other sites, such as the US Embassy, the antenna had to be mounted

on the roof and feedline run several floors to get to the radio and the router. Once the site preparation was complete, it was a very simple matter to install the radio and router.

One major problem developed, once we were over the site preparation issues and got to testing the radios. We found the PCs we selected were unable to have their serial ports run at speeds over 19.2Kbps without data overruns; as soon as we tried to send data to the radios at any speed over 19.2Kbps, nothing worked. After a few days spent in investigating the cause of the problem, we discovered that even though the technical specifications of the PCs indicated that the on-board serial hardware could be run at speeds up to 115.2Kbps, this was not the case. DataCom has some new Pentium PCs that arrived while we were there, intended for a public Internet access center. When we installed Linux on those machines, configured it as a router, and set the serial interface to the 115.2Kbps speed, everything functioned properly.

As luck would have it, we were able to contact someone coming over from the US to Mongolia. This person brought the necessary number of high-speed serial cards over; we were able to install them in the PCs that we had and get them to function properly at the higher datarate. As a result, we were finally able to leave Mongolia, after a slightly longer stay than we'd planned, with the wireless network installed and good connections to the seven remote sites.

The network that we installed is still up and running today. The staff at DataCom has been able to maintain the network and even extend it with additional sites.

Lessons Learned

Here is a summary of some of the key things we learned as a result of this effort:

* A good site survey ahead of installation is the key to the installation's success. The specifics of a survey require people with a good deal of expertise in several areas. If you can't get those people to the site ahead of time to perform the survey, success can be in serious doubt in such remote locations as Mongolia.

* Always make sure that you test the equipment that you're using before you ship it to a remote location. Never believe the specifications for a product until you've tested it yourself! ! !

* Radio is still something like a black art (i.e., magic). There is always something new to be learned. Until we had gone to Mongolia and did this installation, we would have never believed that you could deploy such a wireless network in a dense urban area without having line-of-sight to all locations.

* Low-cost spread-spectrum radio products that are being used in increasing numbers in the US to deliver access to the Internet can be successfully deployed in developing countries with limited expertise in both radio and Internet technologies.

In closing, we'd like to take this opportunity to acknowledge the help and assistance of Steve Goldstein of the NSF, without whose caring and wisdom this effort would not have been possible.

Management of TNCs by means of the Simple Network Management Protocol *

H. Hmida (VA2HLH) and M. Barbeau (VE2BPM)
Département de mathématiques et d'informatique
Université de Sherbrooke
Sherbrooke (Québec), CANADA, J1K 2R1
{hmida,barbeau}@dmi.usherb.ca

Abstract

This article deals with the application of a network management framework, called *Simple Network Munagement Protocol* (SNMP), to manage a particular type of network devices named *Terminal Node Controller* (TNC). TNCs are widely used in the amateur packet radio community. We present new tools based on SNMP for remote management of TNCs. A *Management Information Base* (MIB) has been created for the TNCs parameters we manage in KISS mode. The MIB is implemented under the *Linux* operating system and uses the CMU-SNMP package. We implemented also a new command to manage simultaneously and remotely several TNC parameters.

Keywords: Network management, terminal node controller, management information base, Linux.

1 Introduction

The number of users accessing amateur packet radio is growing from day to day. Managers of telecommunications equipment are called to pay more attention to these pieces of equipment in order to exploit them in a better way. So far, managers were obliged to be close to the packet radio devices they wish to manage and to control. The *Simple Network Management Protocol* (SNMP) [2] is a framework that permits remote management and control of network devices. With a simple design, SNMP has the capability to manage various types of network technologies. For example, SNMP is used for the management of the Internet. The concerns of this paper is the use of SNMP to manage and control wireless networks. In particular, we are interested in the application of SNMP for the control and management of a device type called TNC [3]. A TNC is an important element

*The research described in this paper was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds pour la formation de chercheurs et l'aide à la recherche (FCAR).

of networks of the amateur packet radio community. It is a device placed between a transceiver and a computer implementing a medium access protocol and a data link control protocol.

Classically, management of a TNC can be done using two types of tools, namely, terminal emulation software and utility programs [5]. These tools are helpful as long as the computer on which they run is directly connected to the TNC with a short cable. In other words, these tools are inadequate for remote management of TNCs through networks. The goal of our work is to develop and provide new tools for remote management of TNCs. These types of tools are interesting since the manager that uses them is not obliged to be close to the TNC he wants to manage.

The tools that we propose are based on SNMP. In SNMP, the collection of manageable param eters of a device are logically seen as a data base called a management information base (MIB) [4]. Data elements in a MIB are called objects which are abstractions of real parameters of a device. A MIB offers different services in order to read and modify the values of those objects. SNMP is a protocol that allows use of these services remotely through a network. In order to define the structure of objects in a MIB, the SNMP framework defines a notation called structure of management information (SMI) [1] [2].

In our work, we have identified the TNC parameters that can be managed in KISS mode and defined a MIB for them using the SMI syntax. This MIB is implemented under the Linux operating system and uses the CMU-SNMP package [6]. In addition to the commands coming with this package, we implemented a new command allowing to manage simultaneously and remotely different parameters of a TNC. Those commands access the TNC via SNMP and the MIB we created.

The rest of this paper is structured as follows. Section 2 deals with the basic aspects of SNMP. Section 3 deals with the solution we developed. First, it presents the parameters we manage and the corresponding objects that form our MIB. Second, it explains how to access, read, and set the MIB objects. Finally, it describes the relation between the TNC, the MIB, and how SNMP physically sets the TNC parameters. We conclude with Section 4.

2 Review of SNMP

SNMP was designed as an application-level protocol that is part of the TCP/IP protocol suite (see Figure 1). Four main concepts come with SNMP: management station, agent, management information base, and management protocol. A management station has a set of management applications and an interface by which the network manager may monitor and control a network. An agent responds to requests for information and requests for actions from the management station and may asynchronously provide the management station with important ‘but unsolicited information. It provides the information by accessing its local MIB and retrieving the requested object values. The MIB is a database containing a collection of objects. Each object refers to a network-resource to be managed and is accessed via a data variable. The management station performs the monitoring function by retrieving values oi MIB objects. It can change also values of specific variables. Finally, a network management protocol links the management station and agent. The one used for the management of TCP/IP networks is SNMP.

SNMP is implemented on top of the user datagram protocol (UDP), internet protocol (IP), and the relevant network-dependent protocol (e.g., Ethernet, AX.25). SNMP provides a message exchange mechanism to retrieve and set object values at the agent and notify the management station of significant events. From a management station, three types of SNMP messages are

issued on behalf of a management application: *GetRequest*, *GetNextRequest*, and *SetRequest*. A GetRequest is issued if the management application knows precisely the object it wants to read. A GetNextRequest is issued if the management application wants to read the value of an object that is successor of known object. The GetRequest and the GetNextRequest can only read objects. SetRequest, however, may create or modify objects. These messages are acknowledged by the agent as a *GetResponse* messages, which are passed up to the management application. In addition, an agent may issue *trap* messages in response to events that affect the MIB and underlying managed resources.

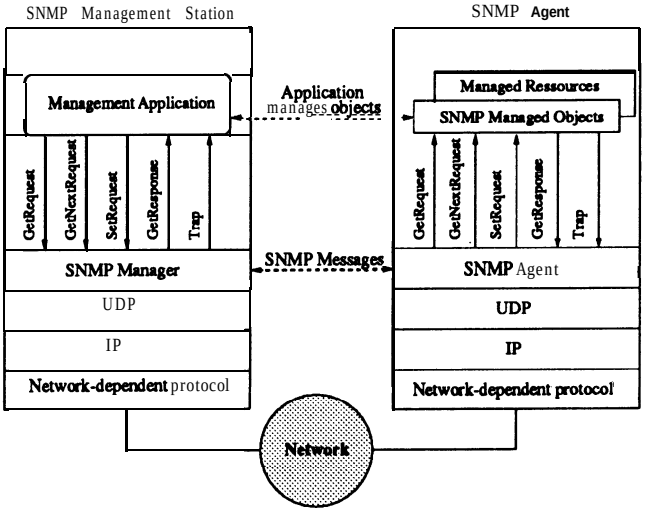


Figure 1: SNMP architecture [1].

3 Development of a solution

In this section, we present a solution we wish propose to solve remote management and control of wireless network devices. The tools we propose are developed under the Linux operating system [8]. Linux is chosen because it has a certain degree of security (like Unix) concerning the access to the system, it supports communication over the AX.25 protocol, and it supports network management. One of the most free popular SNMP packages is CMU-SNMP. It was first designed by the Carnegie Mellon University (http://www.cmu.edu) and then has been ported to Linux by Juergen Schoenwaelder (schoenw@gaertner.de) and Erik Schoenfelder (schoenfr@gaertner.de). This package is fully compliant with the SNMPv1 standard and includes, in addition, some of the new features of SNMPv2. The distribution contains some manager tools that permit, in a command line style, to send requests to devices running SNMP agents. It also contains a SNMP agent program, designed to run under Linux, that provides the manager running on the network (or the same system) information about the status of the interfaces, routing table, uptime, etc. [7]

3.1 Managed parameters and their corresponding objects

In a MIB, objects are organized into groups. Since we are in an experirnental stage, we include our new objects under a group called *experimental group.* Let's now have a look at the objects of a TNC in KISS mode that can be managed and included into a MIB. The KISS mode has a limited number of manageable parameters. They are the following:

- port: the port that is being configured
- fullduplex: sets the TNC into either full duplex or half duplex
- hardware: hardware specific parameters
- txtail: the TX Tail time in milliseconds
- persist: the persist value in milliseconds
- slottime: the slottime in milliseconds
- txdelay: the TX Delay in milliseconds

These TNC parameters, in KISS mode, can be set but their value cannot be retrieved from the TNC. The MIB that we developed allows the management application to get the values given to the parameters. We associated to each TNC parameter a MIB object that reflects its value. For example, the definition of the object associated to the parameter *txtail* is illustrated in Figure 2. The notation is SMI.

```
tncTxTail OBJECT-TYPE
SYNTAX          INTEGER(1..127)
MAX-ACCESS   read-write
STATUS           mandatory
DESCRIPTION
        "Sets the TX Tail time in milliseconds. This value must be
        a multiple of 10."
::= ( tncControl4)
```

Figure 2: Definition of the tncTxTail object.

Figure 2 defines an object called *tncTxTail* situated under the group *tncControl.* The value 4 means that our object is the number four in the *tncControl* group (Figure 3). Also, the definition tells that the value of the object is of type *INTEGER* and can vary between 0 and 127. The label *MAX-ACCESS* defines the way an instance of the object can be accessed (via SNMP or other protocols). In our case, the object is readable and writable (setable). Generally, it can be one of the following: *read-only, read-write, write-only,* or *not-accessible.*

The *STATUS* clause indicates the implementation support required for this object. Support can be one of the following: *mandatory* (the object definition is valid and implementation is required for conformance), *current* (the object definition is valid), *optional* (the object definition is valid,

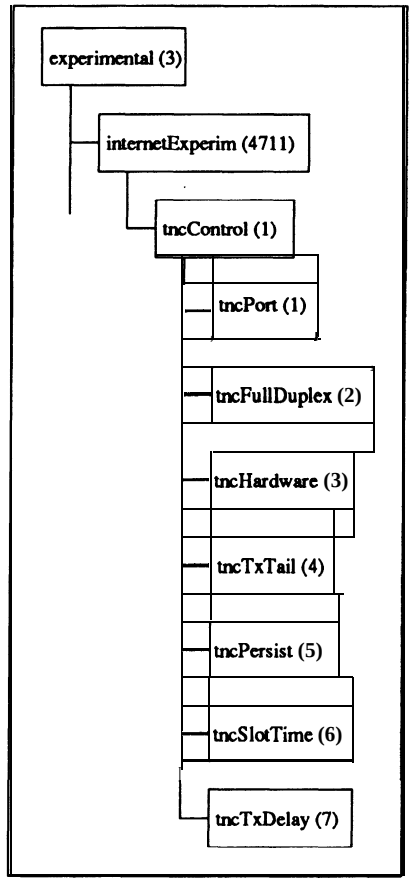


Figure 3: TNC sub-tree.

however implementation is not required for conformance), deprecated (the object definition is valid in limited circumstances, but has been replaced by another), or *obsolete* (managed nodes need no longer to implement this object). A valid definition has the following properties: it is well conceived (precise, unambiguous, complete, and applicable across a wide scope) and relevant (useful and has been used for implementation). With SMIv1, the value *current* is not used. With SMIv2, value *optional* is eliminated and the value *mandato y* is replaced by *current.* The complete SMI definition of all the objects corresponding to the parameters that we manage is given in Appendix 4.

3 . 2 Object access

CMU-SNMP provides three commands for retrieving the values of objects. The first is the *snmpwalk* command. It has three parameters:

- hostname: corresponds to the host name containing the MIB. It can be either *localhost*, if we are addressing our local machine MIB, or the explicit name of a distant host in a textual form (e.g., lolly2.dmi.usherb.ca) or in a numerical form (e.g., 132.210.48.188).
- community name: is one of the community names known by the addressed host. In our case

we use *public*.

- object-ID: is the identifier of the group of objects to be walked. The identifier can be either in a textual form (e.g., system) or in a numerical one (e.g., .1.3.6.1.2.1.1) or a combination of the two (e.g., .1.3.6.1.2.1.system).

To discover, for instance, all the objects under the *tncControl* group, we have just to type:

```
snmpwalk localhost public .1.3.6.1.3.4711
```

a normal response is:

```
internetExperim.tncControl.tncPort.0   =   1
internetExperim.tncControl.tncFullDuplex.0 = 1
internetExperim.tncControl.tncHardware.0   =   3
internetExperim.tncControl.tncTxTail.0   =   20
internetExperim.tncControl.tncPersist.0   =   10
internetExperim.tncControl.tncSlotTime.0   =   10
internetExperim.tncControl.tncTxDelay.0   =   10
```

Another way to obtain the values of these objects is to use the *snmpget* command. To do so, we have to know the exact name of an object. For example, to get the value of the *tncPort* object we can use the following command:

```
snmpget  localhost  public
.1.3.6.1.3.4711.tncControl.tncPort.0
```

a normal returned result is:

```
internetExperim.tncControl.tncPort.0 = 1
```

To know the value of the object next to *tncPort*, we can apply the *snmpgetnext* command in the following way:

```
snmpgetnext  localhost  public
.1.3.6.1.3.4711.tncControl.tncPort.0
```

a normal response is:

```
internetExperim.tncControl.tncFullDuplex.0 = 1
```

This way, we can discover the value of any object within the MIB. To change the value of any writable object, we have just to apply the *snmpset* command. In addition to the hostname, community name, and object-ID, this command needs the type of the object and the new value to be assigned to it. The type can be: *i (INTEGER), s (STRING), x (HEX STRING), d (DECIMAL STRING), n (NULLOBJ), o (OBJID), t (TIMETICKS)* or *a (IPADDRESS)*. Here is an example:

```
snmpset  localhost  public
internetExperim.tncControl.tncSlotTime.0   i   20
```

The *snmpset* command can set only one parameter at a time. To present to the user a better service, we developed a new command, called *snmpax25set*, that permits to set several parameters in a single command line. The syntax of this new command is the following:

```
snmpax25set hostname community object [-p port][-f 0|1][-l txtail][-r persist][-s slot][-t txt]
```

Once the parameters passed to the *snmpax25set* are verified, *snmpset* is called for each one of them.

3.3 Application implementation

Hereafter, we discuss how this application is implemented and how someone can write applications similar to the one we are presenting. We explain in the following lines the C code of the main functions we wrote for this application and the key files necessary for its implementation.

Installation of the CMU-SNMP package for Linux is required. The application uses basically the following three files:

1. *mib.txt*: contains the SMI definition of the MIB objects.

2. *snmp_vars. h:* contains function, constant, and data structure declarations.

3. *snmp_vars.c*: contains the code of the functions necessary to access, read, and set MIB objects.

First of all, the new objects to manage must be added in the appropriate object-group of the MIB. In our case, they are added to the group *tncControl* (a subgroup of *internetExperim*, which is itself a subgroup of *experimental)* (see Figure 3). The part of the file *mib. txt* defining the group *tncControl* is given in Appendix A. The file *mib.txt* also contains predefined standard managed objects. Next, the *tncControl* group need to be registered in order to be known by the agent. The function *snmp_vars_init* in the file *snmp_vars.c* does registration of managed objects. Here are the lines we added to the *snmp_vars_init* function:

```
#ifdef linux
mib,register (
   tnc,id-base,
   sizeof(tnc_id_base) / sizeof(oid), /*num. of obj. id in tnc_id_base*/
   experimental-variables,
   sizeof(experimental_variables)/sizeof(*experimental_variables), /*num. of vars. in the group*/
   sizeof(*experimental_variables)); /*size of a variable description*/
```

tnc_id_base is a vector that defines the identifier of the group *internetExperim* (see Appendix A):

```
#define INTERNET-EXPERIMENTAL 1, 3, 6, 1, 3
static oid tnc_id_base[] =
{
 INTERNET-EXPERIMENTAL,   4711
};
```

experimental_variables is an array that defines all the *tncControl* group objects to manage, their type, access mode, management function, and index under the *internetExperim* group. It is globally defined as follows:

```
struct variable2 experimental_variables[]={
{TNCPORT,       INTEGER, RWRITE, var,experimental, 2, (1, 1}},
(TNCFULLDUPLEX, INTEGER, RWRITE, var-experimental, 2, (1, 2}},
(TNCHARDWARE,   INTEGER, RWRITE, var,experimental, 2, (1, 3}},
{TNCTXTAIL,     INTEGER, RWRITE, var-experimental, 2, (1, 4}},
(TNCPERSIST,    INTEGER, RWRITE, var-experimental, 2, {1, 5}},
(TNCSLOTTIME,   INTEGER, RWRITE, var,experimental, 2, {1, 6}},
(TNCTXDELAY,    INTEGER, RWRITE, var-experimental, 2, (1, 7}}
};
```

TNCPORT, TNCFULLDUPLEX, TNCHARDWARE, TNCTXTAIL, TNCPERSIST, TNCSLOT-TIME, and TNCTXDELAY are defined, in the *snmp_vars.h* file, as follows:

```
#define TNCPORT            1
#define TNCFULLDUPLEX      2
#define TNCHARDWARE        3
#define TNCTXTAIL          4
#define TNCPERSIST         5
#define TNCSLOTTIME        6
#define TNCTXDELAY         7
```

Now, we define new functions that access, read, and set the object values. The main function is *var_experimental*. It is called for each object and is partially defined as:

```
u-char *var_experimental(...)
{
    /* Declaration of local variables */
    ...

    /* Memory allocation for an object */
    ...

    /* switch to object currently handled */
    switch (...){
      case  TNCPORT:
            *write-method  = write_snmp_tnc_Port; (1)
            long-return  = snmp,tnc,port;  (2)
            break;
        /* other cases */
        ...
    3
    return . . . .
3
```

There is a case for each object we are registering. For example, let us consider the case TNC-PORT. There are two actions. First, the function responsible for access to the object is registered, that is *write_snmp_tnc_Port* (1). Second, the initial value of the object is registered, that is *snmp,tnc,port (2). snmp_tnc_port* is a memory variable that *we* define (in file *snmp_vars.c*) and that stores the current value of the managed object.

Function *write_snmp_tnc_Port* does the following:

```
static int write_snmp_tnc_Port(...)
{
   /* Declaration of local variables */
   ...

   /* Assert that the parameter is of type INTEGER */
   ...

   /* Assert that the parameter value is zero */
   ...


      if (action == COMMIT) {
         snmp,tnc,port  =  intval;
         execv("/usr/sbin/kissparms", argv);
```

```
    }
    return  SNMP,ERR,NOERROR;
3
```

If the action performed on the object is a set operation (condition *action == COMMIT* is true), then the global variable *snmp_tnc_port* receives the new value of the object (*intval*) and the *kisspurms* program is called to physically set the TNC parameter.

If the action performed on the object is a get operation, then the agent retrieves and returns the value in the memory variable *snmp_tnc_port*.

4 Conclusion

In this paper, we have presented a method to manage a particular type of network devices, named TNC, by means of a network management framework called SNMP. Precisely, we have implemented a MIB containing the parameters we intend to manage (Section 3.1). Those parameters are seen as objects that can be read and set using simple commands (Section 3.2). To facilitate the TNC management, we have implemented a new command that sets remotely and simultaneously several parameters. In Section 3.3, we have explained how we used the CMU-SNMP package to implement our application. [1]

Two problems were uncovered by this work. First, we noted that in Kiss mode, there are no ways to read and verify the actual values of TNC parameters. Consequently, the strength of the solution we propose depends on the reliability of the *kisspurms* program we are applying to physically set TNC parameters. Second, there may be an inconsistency between MIB memory variables and actual TNC parameters. In fact, if the agent computer shuts down then the content of memory variables and updates of those variables are lost. When the computer reboots, these memory variables take default values which do not necessarily correspond to the real values in the TNC.

Actually, there are two possible solutions to this problem. First, values of memory variables are maintained in files. Updates are both reflected in the memory variables and in the file. When the system reboots, initial values are taken from that file. Second, when get requests are received, instead of reading values from memory variables, a procedure directly accesses the values in the TNC. The second solution is actually not feasible with the actual definition of the Kiss protocol.

[1] A copy of the C code of this project can be found at http://www.dmi.usherb.ca/~hmida/project.html.

References

[1] William Stallings, "*SNMP, SNMPv2, and CMIP. The Practical Guide to Network-Management Standards"*, Addison Wesley, 1993.

[2] Marshall T. Rose, *"The Simple Book. An Introduction to Internet Management"*, Prentice Hall, Second Edition, 1994.

[3] Philip R. Karn, Harold E. Price, and Robert J. Diersing, *"Packet Radio in the Amateur Service"*, IEEE journal on selected areas in communications, Vol. SAC-3, NO 3, May 1985, pp. 431-439.

[4] David Perkins and Evan McGinnis, " *Understanding SNMP MIBs*" , Prentice Hall, 1997.

[5] Terry Dawson, "*Linux AX25-HO WTO, Amateur Radio"*, http://www.sunsite.unc.edu/mdw/HOWTO/AX25-HOWTO.html

[6] Erik Schönfelder, "*Linux CMU SNMP Project"*, http://www.gaertner.de/snmp/

[7] David Guerrero, *"Network Management & Monitoring with Linux*", Linux Journal, June 1997, pp. 36-44.

[8] " *Linux Home Page"*, http://www.linux.org

Appendix A: Definition of the tncControl group and its objects

```
experimental
   OBJECT  IDENTIFIER ::= { internet 3 3
internetExperim
   OBJECT IDENTIFIER ::= { experimental 4711 }
tncControl
   OBJECT  IDENTIFIER ::= { internetExperim 1 }

tncPort OBJECT-TYPE
   SYNTAX      INTEGER(1..127)
   MAX-ACCESS  read-write
   STATUS      current
   DESCRIPTION
     "The port name that is being configured."
   ::= { tncControl 1 3

tncFullDuplexOBJECT-TYPE
   SYNTAX      INTEGER(1..127)
   MAX-ACCESS  read-write
   STATUS      current
   DESCRIPTION
     "The TNC can be set into either full duplex
```

```
      or half duplex."
    ::= { tncControl 2 }

tncHardware OBJECT-TYPE
    SYNTAX      INTEGER (1..127)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
      "This is used to set the hardware specific
      parameters."
    ::= { tncControl 3 }

tncTxTail OBJECT-TYPE
    SYNTAX      INTEGER (1..127)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
      "This is used to set the Tx Tail time in
      milliseconds.
      Only values like 10, 20, etc are used."
    ::= { tncControl 4 }

tncPersist OBJECT-TYPE
    SYNTAX      INTEGER (1..127)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
      "This is used to set the persist value.
      This parameter is scaled to the range
      0 to 255."
    ::= { tncControl 5 }

tncSlotTime OBJECT-TYPE
    SYNTAX      INTEGER (1..127)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
      "This is used to set the slottime time
      in milliseconds. Only values like 10,
      20, etc are used."
    ::= { tncControl 6 }

tncTxDelay OBJECT-TYPE
    SYNTAX      INTEGER (1..127)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
      "This is used to set the TX Delay time
      in milliseconds. Only values like 10,
      20, etc are used."
    ::= { tncControl 7 }
```

North American Digital Systems Directory (NADSD)

Greg Jones, WD5IVD
President, TAPR
8987-309 E. Tanque Verde #337
Tucson, AZ 85749-9399
wd5ivd@tapr.org

Carl Estey, WAOCQG
NADSD Manager
10021 Drew Ave South
Bloomington, MN 55431
wa0cqg@tapr.org

Abstract

Have you ever wanted to know if there might be a Packet BBS in a distant city where a friend lives? Or what the frequency is of the PacketCluster station in your area? Many times it isn't easy to find out about digital services in a distant area. In the past, one way to get this information was to consult the packet listings in the American Radio Relay League (ARRL) Repeater Directory. That's now a thing of the past. The North American Digital Systems Directory (NADSD) project was begun in January of 1997 to make information concerning amateur radio digital systems available to amateur radio operators. This paper will describe the history, purpose, and functions of the NADSD.

History / Purpose

During the last few months of 1996, while planning the 1997-98 edition of the ARRL Repeater Directory, the ARRL concluded that the Repeater Directory was no longer the most effective medium for information regarding amateur radio digital systems. This was a mild shock to many in the digital community, but instead of seeing doom and gloom, several regional digital groups approached TAPR shortly after the announcement and asked if something could be done to salvage the situation. Discussions involving various regional digital groups that provide data to the digital section of the ARRL Repeater Directory and the staff at ARRL HQ led to the conclusion that TAPR was the logical group to take on the task. TAPR President Greg Jones, WD5IVD, and a team of volunteers representing a number of regional groups quickly defined and developed an all-electronic World-Wide-Web based replacement, known as the North American Digital Systems Directory (NADSD) <http://www.tapr.org/directory>.

With the support of the American Radio Relay League (ARRL), TAPR began creating and organizing the NADSD project. Within 3 weeks of the ARRL announcement about the removal of the digital section from the Repeater Directory, the TAPR development team had been formed and three weeks later the first version of the NADSD was operational with several minor additions taking place over the next few months.

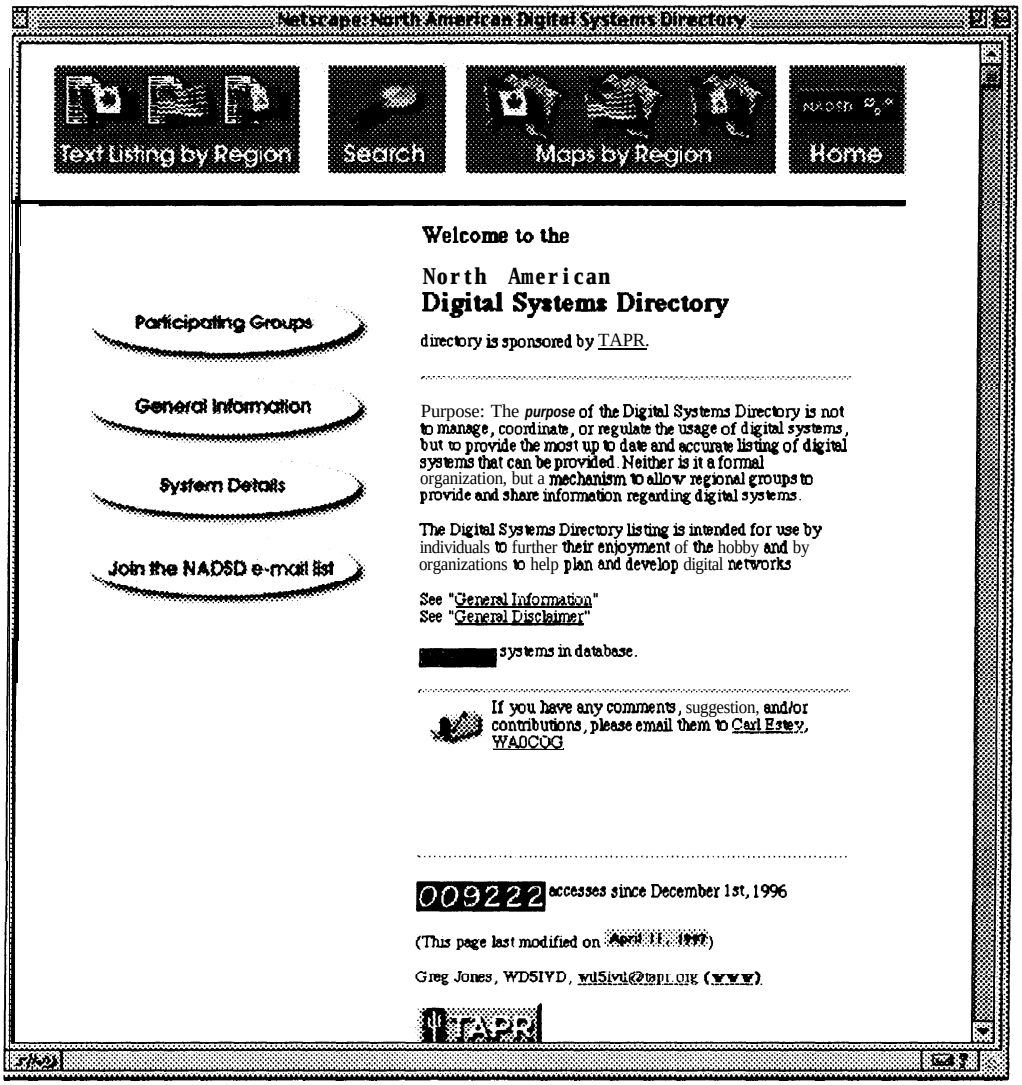


Figure 1 - NADSD Home Page

The NADSD describes systems used by amateur radio stations involved in digital communications in United States, Canada, and Mexico. The Digital System Directory is based on information provided by regional, state, and local organizations as well as individuals in a nearly realtime format. This allows information to be maintained and updated more frequently than in a yearly publication. TAPR has made and will continue to make the NADSD database available on TAPR's yearly CD-ROM and is looking into doing some type of printed document in the future for those without technology access at home.

The purpose of the Digital System Directory is not to manage, coordinate, or regulate the usage of digital systems, but to provide the most up-to-date and accurate listing of digital systems that can be provided. Neither is it a formal organization, but a mechanism to allow regional groups to provide and share information regarding digital systems. The Digital System Directory is simply intended for use by individuals to further their enjoyment of the hobby and by organizations to help plan and develop digital networks.

Participation / Activity

The NADSD began with a short list of regional groups, but soon grew to represent over ninety information submitters (regional groups, local groups, and individuals). Figure 2 shows the growth of the system and compares it to the Digital Listing of the 1996-97 ARRL Repeater Directory As of August 1997, eight months into the project, the NADSD is over 56% larger then the last Repeater Directory Digital Listing. 4175 systems are present in the NADSD as of the middle of August, covering forty-two states of the US and eight Canadian Provinces. Figure 3 gives a current breakdown of activity by state / province. Several states are still in the 'area to develop' category. If you are in one of these areas, you might check on who was providing this information in the past and have them contact the NADSD.

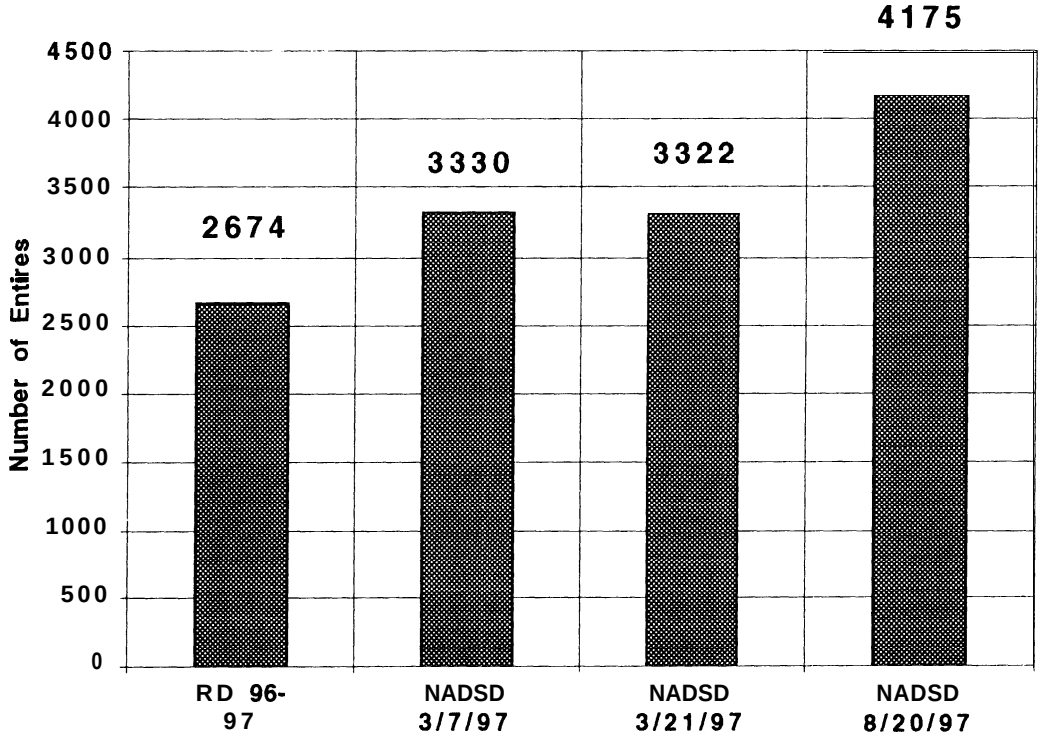


Figure 2 — NADSD Growth

New groups or individuals wishing to submit data to the directory need to fill out an application that will be sent to the overview committee for consideration. To keep the directory data gathering process as simple as possible, we ask for people to check the list of Participating Organizations first to see if someone is not already covering their area. Working with one of these existing groups is preferable to having two groups each submitting data on the same systems. However, like anything in amateur radio, the system has several groups that overlap and the system has no problem handling this issue. If you feel that your group can contribute new accurate data on systems not being covered at the present time, then we welcome your application.

If you have a suggested change or error correction to a record in the directory, please contact the person submitting the data for that record entry. The name should be found at the end of the record data row and can be clicked on to create an e-mail message.

Participating Organizations, after their applications are accepted, are given explicit instructions by private e-mail on how and where to upload their data files. They are given a password to allow secure access to their files in the database directory. This way, each data submitter/group has their own private upload area, to which no one else has access, to view the raw upload data.

Participating Organizations should upload new data records into the system whenever the system information has changed. The automatic directory update program runs at regular intervals and if it detects an updated file, it generates a new web page based on that revised information.

	Rpt Dir 96-97	NADSD 3/7/97	NADSD 3/21/97	NADSD 8/20/97	Varience from last period	Varience from 96-97 Rpt Dir
TOTAL # OF LISTINGS	2674	3330	3322	4175	853	1501

Indicates area to develop

Present Growth compared to RD 56.1%

STATE/PROV.	Rpt Dir 96-97	NADSD 3/7/97	NADSD 3/21/97	NADSD 8/20/97	Varience from last period	Varience from 96-97 Rpt Dir
Alaska	47	42	42	40	-2	-7
Alabama		29	31	29	-2	29
Arizona		83	43	44	1	44
Arkansas	106	31	31	31	0	-75
California	138	225	225	234	9	96
Colorado	129	2	2	7	5	-122
Connecticut	7	50	48	54	6	47
Delaware	9	21	21	21	0	12
Florida	129	169	169	173	4	44
Georgia		1	4	3	-1	3
Hawaii	9				0	-9
Idaho	14			1	1	-13
Illinois	211	236	238	237	-1	26
Indiana		230	234	234	0	234
Iowa	5	2	2	52	50	47
Kansas	94	14	15	13	-2	-81
Kentucky					0	0

Figure 3 — NADSD Activity by State/Region

Louisiana	57	25	25	25	0	-32
Maine	25	29	29	35	6	10
Maryland	60	140	140	140	0	80
Massachusetts	13	82	85	98	13	85
Michigan	155			148	148	-7
Minnesota	20	52	52	52	0	32
Mississippi		1	1	4	3	4
Missouri		4	7	7	0	7
Montana	57				0	-57
Nebraska	21			13	13	-8
Nevada	67	21	24	43	19	-24
New Hampshire	33	39	39	46	7	13
New Jersey	28				0	-28
New Mexico	28	36	36	36	0	8
New York	95	21	21	26	5	-69
North Carolina	137	207	203	203	0	66
North Dakota	37				0	-37
Ohio	66	63	63	68	5	2
Oklahoma		73	82	69	-13	69
Oregon	110	118	119	125	6	15
Pennsylvania	197	121	126	154	28	-43
Rhode Island	24	29	29	30	1	6
South Carolina		58	57	55	-2	55
South Dakota					0	0
Tennessee					0	0
Texas		539	538	550	12	550
Utah	117	1	1	1	0	-116
Vermont	10	16	18	18	0	8
Virginia	17	57	57	224	167	207
Washington	71			249	249	178
West Virginia	58	123	123	121	-2	63
Wisconsin	85	193	193	193	0	108
Wyoming	23				0	-23
# of listings	**2509**	**3183**	**3173**	**3906**	**733**	**1397**
# of States	**38**	**38**	**38**	42	4	4

Figure 3 — NADSD Activity by State/Region (cont.)

STATEIPROV.	Rpt Dir 96-97	NADSD 3/7/97	NADSD 3/21/97	NADSD 8/20/97	Varience from last period	Varience from 96-97 Rpt Dir
Alberta	43					-43
British Columbia	13			41	41	28
Manitoba	11					-11
Maritime	7	7	7		-7	-7
New Brunswick				12	12	12
Newfoundland						
Northwest Territory	1					-1
Nova Scotia				40	40	40
Ontario		22	25	23	-2	23
Prince Edward Island				8	8	8
Quebec	44	93	93	93		49
Saskatchewan	4	12	12	37	25	33
Yukon	4	13	12	15	3	11
# of listings	127	**147**	**149**	269	**120**	**142**
# of Prov/Terr.	8	**5**	**5**	8	**3**	**0**

STATEIPROV.	Rpt Dir 96-97	NADSD 3/7/97	NADSD 3/21/97	NADSD 8/20/97	Varience from last period	Varience from 96-97 Rpt Dir
Puerto Rico	6	0	0	0	**0**	-6
U.S. Virgin Islands	3	0	0	0	**0**	-3
Br. Virgin Islands	1	0	0	0	**0**	-1
Costa Rica	4	0	0	0	**0**	-4
Cuba	3	0	0	0	**0**	-3
Mexico	10	0	0	0	0	-10
Peru	11	0	0	0	0	-11
# of listings	38	**0**	**0**	**0**	0	-38
# of Countries/Terr.	7	**0**	**0**	**0**	**0**	-7

Figure 3 — NADSD Activity by State/Region (cont.)

The NADSD is designed to automatically run and maintain itself with little or no intervention by the project team. Participating groups simply upload their updated information and every several hours the system checks all data to see if anything has changed. If changed data is present, the system generates all new pages to reflect the changes made by the submitting group(s). In this way, the NADSD is a very decentralized structure and allows groups to update and maintain their information as it is convenient for them to do. Some groups have updated their information weekly, others monthly, and a few others just the initial time. It is up to the participants to choose the frequency in which they update their information. In this way, the project team has no other worry than to make sure the system is running correctly, or if a group uploads poorly formatted data to work with them, to get the format corrected. Helping groups understand the format, formatting, and ftp uploading has been one of the largest chores during the project. Now that the initial groups have been brought up to speed on this process, the load has lightened, with only new groups being worked with.

As contrasted with the manner in which the ARRL Repeater Directory was done, the NADSD automated system allows the project team to now focus more on long-range issues of quality, area of coverage, and future information distribution, instead of having to process the data by hand each year.

The NADSD provides data in two formats based from the data submitted: Text output and Maps. The Text output is presented in both HTML and as a text file that can be downloaded and posted off to packet radio or other means if necessary. The mapping, which uses Steve Dimse's, K4HG javAPRS software, allows for areas to be looked at in a graphical fashion. The ability to look at the data in both text and mapped format is a huge step forward from previous methods of information distribution. The text and maps are available for selection off the home page using the menu bar on top. On the left, the text sections are situated by area and on the right, the mapping sections can be selected.

The Text Listing

When a user first enters the text section, they receive a map to the area that can then allow a selection of a specific region or section. Figure 4 shows the Canada selection page. The user would then choose a province within Canada to see that text listing. The text listing provides Output and Input Frequencies, Call and alias, System Information including speed, type, software, IP, etc, with location and information on the sponsoring organization and who submitted it. A link is provided at the end of each entry, so corrections can be sent directly to the submitter of the information.

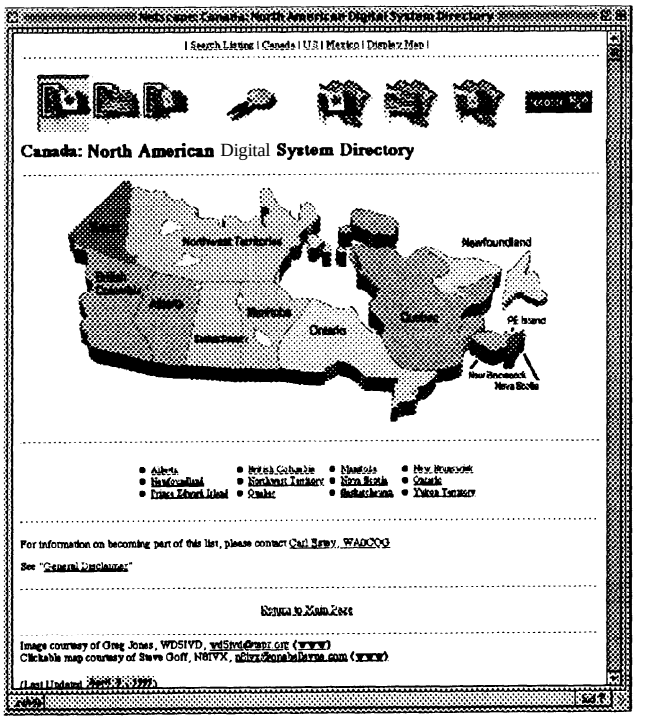


Figure 4 - Text Selection** for **Canada

Ontario: North American Digital System Directory

OUTPUT INPUT	CALL ALIAS	SPEED SYSTEM TYPE NETWORK TYPE SOFTWARE USED IP ADDRESS FQDN	COUNTY LOCATION GEOGRAPHIC AREA	SPONSOR & ORGANIZATION NOTES	SUBMITTER LAST UPDATED
Coldwater					
440.025	VE3FJB-5 MEDONTE	1200 Switch BPQ FN04eq	Simcoe Medonte Simcoe	VE3FJB HEX-9 Emerg. Pwr.	A. Mitchell 18-08-97
430.55	VE3FJB-5 MEDONTE	19200 Switch BPQ FN04eq	Simcoe Medonte Simcoe	VE3FJB HEX-9 Emerg. Pwr.	A. Mitchell 02-07-1997
Dwight					
145.01	VE3MUS-1 MSKOKA	1200 Switch NetROM	Muskoka Dwight Muskoka	VE3KR MFMC	A. Mitchell 02-08-1997
440.025	VE3MUS-11 #LKBYS	1200 Switch NetROM	Muskoka Dwight Muskoka	VE3KR MFMC	A. Mitchell 02-08-1997
Keswick					
445.95	VE3YRA-3 NWMRKT	1200 Switch BPQ	York Keswick York	VE3CGR YRARC	A. Mitchell 18-08-97
445.95	VE3YRA-3 NWMRKT	9600 Switch BPQ	York Keswick York	VE3CGR YRARC	A. Mitchell 18-08-97
North Bay					
145.01	VE3NBC NORBAY	1200 Node NetROM	Nippissing North Bay Nippissing	VA3PC n/p	A. Mitchell 02-08-1997
Owen Sound					
436.3	VE3XOX-10 #XOXHS	19200 Switch BPQ	Grey Owen Sound Grey	VE3XOX n/p	A Mitchell 02-07-1997
430.55	VE3XOX-9 OS445	1200 Switch TheNET	Grey Owen Sound Grey	VE3XOX n/p	A. Mitchell 03-16-1997
Parry Sound					
440.025	VE3PSP-3	1200 Switch BPQ	Parry Sound Parry Sound Parry Sound	VE3LWC n/p	A. Mitchell 18-08-1997
Tory Hill					
145.01	VE3TBF TORHIL	1200 Switch NetROM	Haliburton Tory Hill Haliburton	n/p	A. Mitchell 02-08-1997

Figure 5 - Ontario Text Listing

One of the newest features of the system is the inclusion of dynamic maps. The user can select the mapping icons to see various selections for each country. Figure 6 shows the selection page for the US. Once a section has been selected, the user will receive a javAPRS screen. Figure 7 shows the one for Texas. The user has complete control over what is displayed and the zoom.

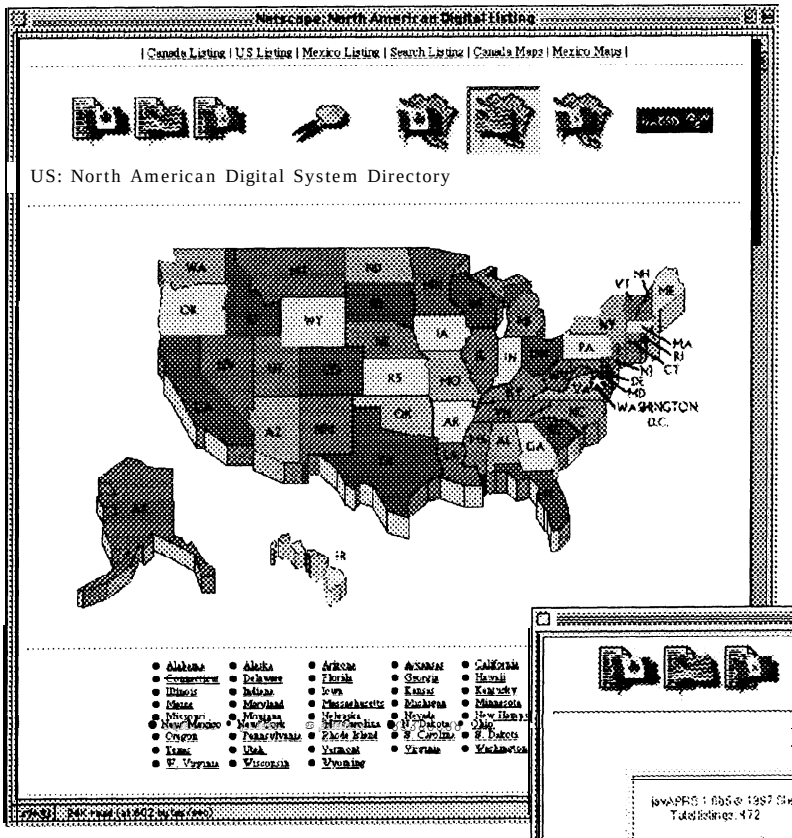


Figure 6 - Map Selection for US

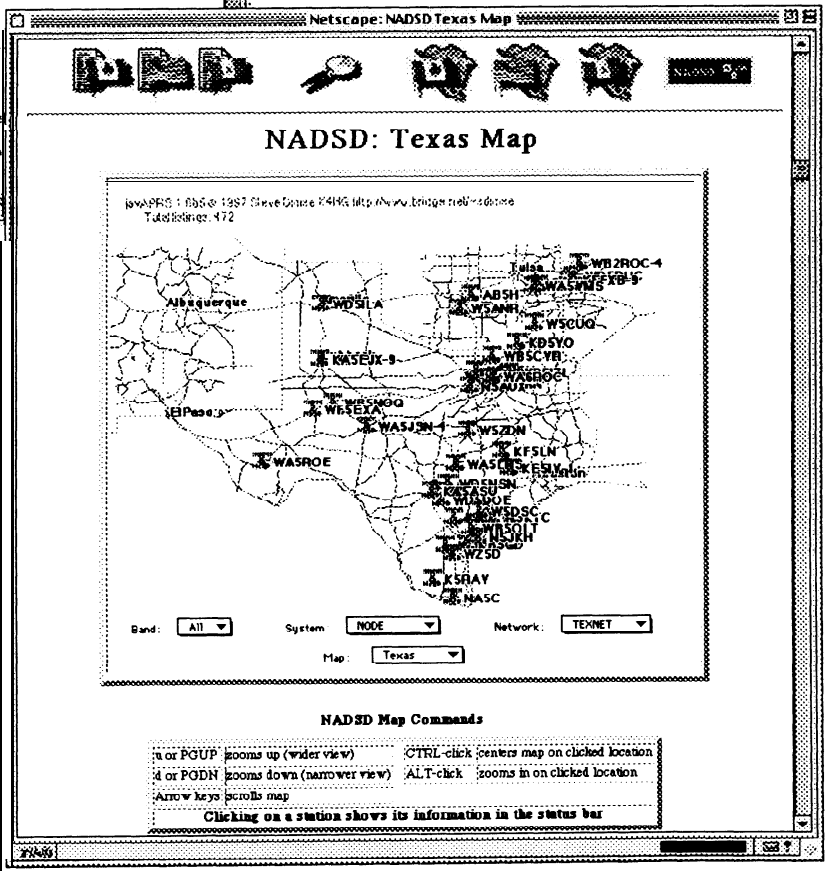


Figure 7 - Map of Texas and surrounding states

Figure 8 shows a zoom in on the Dallas/Ft Worth area showing only TexNet nodes. The ability to control and see visually where nodes are located in relation to one another is a tremendous tool of the system. In the next version, we are looking at ways for the data submitters to show links between systems, in order to show networking interconnects.

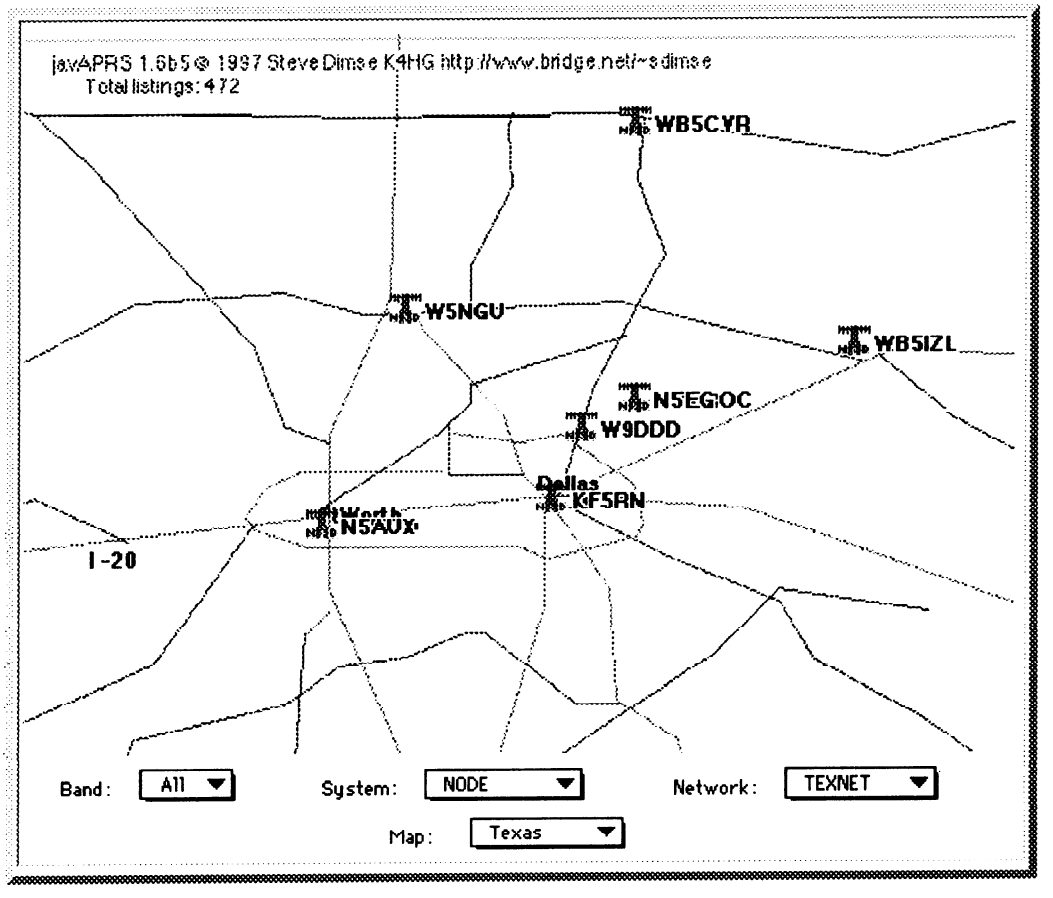


Figure 8 -Zoom in on Dallas/Ft Worth

Data Entry Method

The 1997 format of the Directory is an expanded format based upon the older format used by the ARRL Repeater Directory. The reason for this was twofold 1) data submitted by the ARRL was easier to incorporate into the system and 2) participating groups were able to expand their own ARRL databases during the first year.

The data file submitted is an ASCII Tab Delimited file format. The only fields that are mandatory for year one are STATE (field D) and CITY (field U). Submitters have been supplying most of the fields which has helped the quality of the data greatly. Figure 9 shows the database file definitions.

Figure 9 - NADSD Data Format

Field	Field Name	Permitted Entriesor & Examples	Explanation
A	BAND	3.5 I 7 I 10 I 14 I 18 I 21 I 24 I 29 I 52 I 144 I 219 I 222 I 440 I 1200 I ...	Enter the nominal band frequency as one of the indicated values. Enter it exactly as shown. If your system does not match any of these "preset" entries, contact the committee for guidance.
B	OUTPUT	145.01	Enter the primary output Frequency of the system expressed in MHz. For channelized operation (VHF/UHF), enter the nominal channel frequency such as the examples shown. For HF operation , enter the exact "mark" frequency for the mode of operation being used.
C	INPUT	446.025 446.025	For any system in which the input frequency is different from the output frequency (field B), enter the input frequency here. For VHF/UHF channelized operation, enter the nominal channel frequency such as the example shown. This field would apply to all types of repeaters (with or without attached nodes), as well as "split frequency" simplex and full duplex nodes. Please enter the details of the system in the NOTES field (field K).
D	STATE	**USA:** CT I CA I TX I ... CANADA: AB I BC I MB I NB I NF I NS I NT I ON I PE I QC I SK I YT **MEXICO:** TBD	Enter the Two Letter State Designator as defined by the US Postal Service, the 2 letter Canadian Province Designator as defined by Canada Post or the Mexican state designator shown. Please note that for the Canadian province of Quebec is listed as QC, but the system will also accept PQ as an alternate. Please note that for the Canadian province of Yukon Terr. is listed as YT, but the system will also accept YK as an alternate. The STATE field is necessary for the proper operation of the automatic WEB generating program and the 2 letter code must be entered exactly as shown. Otherwise the record will be ignored.
E	LOCATION	Upper Podunk	Enter the name of the nearest &own or city to the exact location of the system. Where the system is wide spread such as APRS, enter an appropriate name for the whole area that the system covers such as "Statewide". Data should be in word capital format (i.e. Location).
F	CALL	WA1XYZ-5 WD6ABC	Enter the callsign and SSID of the System. Where the SSID is 0, it should be omitted.
G	SPONSOR	WIARC/VE2CWI	Enter the name or initials of the Sponsoring group, club or individual and the callsign normally used by that club or person. If both an individual and a group sponsor the system, then enter the individual's name here and enter the group's name in field Y (ORGANIZATION).
H	LAST UPDATE	02-21-1997	Enter the date when the information of the entry was last updated. As the entries information is updated or corrected, this date should reflect the date the submitter changed the information in their database. It is to be expressed in the numerical format MM-DD-YYYY .
I	SOURCE	UNY REPCO TPRS	Enter the name or initials of the group that the person submitting the data represents. Also known as the "Participating Organization".
J	GEOG AREA	NW VTSo CAE ON	Enter the geographic location (within the state) or the county area within which the system operates. County may be abbreviated as "cty". Data should be in word capital format (i.e. Geog Area).

K	**NOTES**	On Mt. xxx, Btty Power, Bit regen rptr, etc	Enter any explanatory notes that would indicate unusual features of the system and that would help the user better understand how to use the system. Please keep it short. Abbreviate if necessary. If the system is a backbone link, indicate here that the system is not for User access. If the system is a "network server" (not having a local user port), then indicate here the nearest network user port and connect path to that network server.
L	**ALIAS**	ARA8, SRTOGAGABLES, #2NUUE, etc	Enter the Node Alias assigned to the system.
M	**SPEED**	1200 I 2400 I 4800 I 9600 I 19200I ...	Enter the data rate of the system in bits per second (bps). The value should be entered exactly as the numbers shown. If the system operates at a data rate different from these "preset" numbers, contact the committee for guidance.
N	**SYSTEM TYPE**	SWITCH I DIG1 I BBS I NODE I DX CLUSTER I APRS I DIGITAL RPT I LINK I	Enter the closest term that defines the system type. If the system does not fall exactly into one of these terms, enter the closest term and explain any differences in the notes (field K). For example: an audio repeater for packet use would be a Digital Repeater with the Audio type noted in the notes. Be careful to enter the term exactly as shown, as the SYSTEM TYPE string is used to define the map icon used in the automatic NADSD mapping system. If your system does not match any of these "preset" entries, contact the committee for guidance.
0	**NETWORK TVPE**	ROSE I NETROM I THENET I BPQ ITCP/IP I TEXNET I APPLETALK I GATEWAY I X1J I PC/TNC I FLEXNET I KANODE I	Enter the name of the networking protocol used for L3/L4 network systems. Be careful to enter the term exactly as shown, as the NETWORK TYPE string is used to define the map icon used in the automatic NADSD mapping system. If your system does not match any of these "preset" entries, contact the committee for guidance.
P	**SOFTWARE USED**	FBB I RLI I MSYS I AA4RE I AK1A I JNOS I ...	Enter the name of the software used for the network or server system. Please enter the terms specified exactly as shown since they are used by the automatic programs.
Q	**IP Address**	129.120.111.78	Enter the full numeric Internet address of the system (if applicable). It is entered in the format ###.###.###.### with each numerical group separated by decimal points. This field would normally only be used by systems used as TCP/IP routers or servers.
R	**FQDN**	wb1dsw.ampr.org	Enter the "Fully Qualified Domain Name" of the system. Like the IP address, this field would only be used by systems used as TCP/IP routers or servers. The format is normally a number of lower case text strings separated by decimal points. Be very careful to observe the correct case (upper or lower case) when entering the FQDN
S	**LAT**	4515.48	Enter the Latitude of the system in the format <degrees minutes>.<hundredths of minutes> (DDMM.HH). North latitude is assumed. If your available data is in degrees/minutes/seconds, then the seconds can be converted to hundredths of minutes by multiplying by 1.66 . If your available data is in the "Maidenhead Locator" format (eg FN34be) then there are programs available to convert it to degrees/minutes.hundredths. One example of such a program is available on most F6FBB BBS systems. Note that the Latitude field entry is essential for the system to be entered into the NADSD mapping system. If you do not wish to give the exact location of the system, then round the latitude value off to the nearest minute (1 mile) or 10 minutes (10 miles).

T	**LON**	7529.12	Enter the Longitude of the system in the format <degrees minutes>.<hundredths of minutes> (DDDMM.HH). West longitude is assumed. If your available data is in degrees/minutes/seconds, then the seconds can be converted to hundredths of minutes by multiplying by 1.66 . If your available data is in the "Maidenhead Locator" format (eg FN34be) then there are programs available to convert it to degreeslminutes.hundredths. One example of such a program is available on most F6FBB BBS systems. Note that the Longitude field entry is essential for the system data to be used by the NADSD mapping system. If you do not wish to give the exact location of the system, then round the longitude value off to the nearest minute (0.5-l mile) or 10 minutes (5-10 miles)
U	**CITY**	Syracuse San Diego Lower Podunk	Enter the name of the nearest major town or city which the system serves. The directory is indexed by this field within each state and this field is mandatory. The data may be the same as LOCATION (field E). Data should be in word capital format (i.e. City Name). The CITY field is necessary for the proper operation of the automatic WEB generating program. If this field is blank, the record will be ignored.
V	**COUNTY**	OTSEGO ORANGE etc	Enter the name of the state County that the system is physically located in. The word county is not needed. Data should be in word capital format (i.e. County Name).
W	**STATE**	CA I TX I PQ I etc ...	Enter the US or Mexican state or the Canadian Province that the system is located in. The format is the official 2 letter postal designation. This entry is the same as field D defined above and that field can simply be copied over to this field.
X	**COUNTRY**	USA, Canada, or Mexico,	Enter the name of the country in which the system is located. This field determines the database group into which the data record will be entered.
Y	**ORGANIZATION**	LPARC Lower Podunk ARC	In cases where there is more than one sponsor for the system, enter the second name here. The primary sponsoring group or the individual sponsor is entered in. field G.
Z	**SUBMITTER INFO**	Joe Ham, N9XYZ	Enter the name and callsign of the person submitting the data to the database. This name is displayed at the end of the record file so that those people who have changes or comments can forward the necessary information to the person who has the password priviledge for modifying the files in question.
AA	**SUBMITTER E-MAIL**	jham @insane.net	Enter the Internet e-mail address of the person submitting the data files to the directory database. This address is used to automatically direct any changes/comments to the person responsible for the data.
AB	***SOURCE* E-MAIL**	organize@insane.net	Enter the Internet e-mail address for the "Participating Organization" who the person submitting data to the database represents. The name of this organization is entered in field I.

The Microsoft's Excel spreadsheet program is a very good way to create the required "TAB delimited text" file. Submitters can either enter raw information directly into the spreadsheet cells or import data from another database program such as dBase IV The spreadsheet format allows easy filing of whole columns with the same value (such as the "Country" field) and of moving whole columns around. Full featured word processors can be very useful to filter text files for unwanted characters, etc. When dealing with long lines of text without carriage returns, most word processors will automatically word wrap the text. This makes it very difficult to distinguish between separate records. If possible, turn off the word wrap feature or have the processor insert a visible symbol for the carriage return. Some simple text editors do save TABs intact when you save an edited file. Others do not, but rather convert TABs to spaces. Norton Editor, MS WRITE, and MS NOTEPAD have been tested and found to export the TABs intact. MS EDIT and UED (a simple DOS shareware text editor) do NOT save TABs but rather convert them to spaces. There are many utility programs that can be used to view the real contents of ASCII files. Hex editors such as PCTools hex editor or Norton Disk Editor will work well if you have them. LIST is a simple DOS shareware program that is very good. It will view ASCII files and display them with or without visible control character symbols or as a hex editor.

How the System Works

The functions of the system are fairly straightforward and are comprised of two phases. Phase 1 allows the submitter to upload and then have the data checked and Phase 2 rebuilds the web pages and maps.

Phase 1 - Submitter Upload / CHECKPROGRAM

Each submitter is assigned an FTP area to uploaded their information files into. Each hour the CHECKPROGRAM runs on the TAPR.ORG system and checks for any changes in the files. If a file has changed, the CHECKPROGRAM then examines the new input and e-mails a report back to the submitter regarding the status of the latest upload. This checking routine has been very handy in allowing quick feedback to the submitter and getting changes reuploaded to correct information, problems, or errors in the data provided. In addition, the CHECKPROGRAM uses the GIS database of locations to help submitters that have left blank the LAT/LON fields and suggest possible location information. This has helped to get initial data within a geographic area and made available to the mapping part of the NADSD until much more accurate LAT/LON information can be collected. The GIS database looks something like this for Austin, Texas: Austin, ppl, TX, Travis, 48, 453, 3016.01N, 09744.34W, BGN 1931,501, Austin East, 514013.

Phase 2 - GENPAGES

Phase two is the creation of the text information web pages, plain text output, and data that javAPRS uses. This is accomplished using a PERL script developed by the author and takes about 15 minutes to run with the over 4000 entries in the system. GENPAGES reads in each data file in the system and then generates all the necessary HTML code, text output, and data for javAPRS to use. The GENPAGES program run every 4 hours which allows the NADSD to be updated six time daily as new and updated information is made available to the system.

The Future / Conclusion

The NADSD is approaching the conclusion of year one and will begin to work towards making sure all groups double check their information for the upcoming CD-ROM distribution. Some of the issues we had to face during the first year was related to education regarding how to get each user up to speed on making their information available in TAB delimited ASCII output uploaded to TAPR.ORG via ftp sessions. For some this was not an issue, but for many more there was a step learning curve involved with some amount of frustration at the new environment that the system was being maintained in. After 6 months of operations, most of these problems have subsided and the task ahead is to focus on better data and more coverage.

There has been concern voiced by some that the 'packet radio' data is not being made available via packet radio. Many of the concerns focus on the thought that the Internet is taking away from packet radio. This was one reason the text version of the output was generated, but the problem now is that many of the files are just to large to be placed on the packet radio networks and packet radio BBSs. We hope having the information published once a year in the TAPR CD-ROM and also the possibility of some limited amount of printed materials will help offset the lack of distribution over the packet radio networks and packet radio BBSs for the actual systems that are being cataloged. If anything, the NADSD is a good example of how the Internet is helping to provide a service that would never have been possible over the current Packet Radio network. Use of the NADSD as a tool to find out about sites in your area, or for an upcoming site in a quick and effective manner, should mean more usage of packet radio systems that are active.

The future is bright for the NADSD, but like any volunteer project,, the group is always looking for others that would want to help out. We need to find a few individuals that want to take up the chore of beating the bushes to get more data submitters in areas that we are lacking, as well as help in writing articles about the NADSD to help spread the word about the system.

For further information on the project and how to get involved, regional groups should check http:/ /www.tapr.org/directory or send e-mail to Carl Estey, wa0cqg@tapr.org

References

1. TAPR's North American Digital Systems Directory (1997a). Information [1 web pages]. Available Web: http:/ /www.tapr.org/directory/info.html

2. TAPR's North American Digital Systems Directory. (1997a). How it Works [1 web pages]. Available Web: http://www.tapr.org/directory/how.html

3. TAPR's North American Digital Systems Directory (1997c). NADSD Home Page Available Web: http:/ /www.tapr.org/directory/

4. TAPR, 8987-309 E. Tanque Verde Rd #337, Tucson, AZ, 85749. Office: 940-383-0000. Fax: 940-566-2544. E-mail: tapr@tapr.org. Web: http: / /www. tapr.org

TAPR Status Report on Spread Spectrum Activity in the Amateur Radio Service

Greg Jones, WD5IVD
President, TAPR
8986-309 E. Tanque Verde Rd #337
Tucson, Az 85749-9399
wd5ivd@tapr.org

Dewayne Hendricks, WA8DZP
Chair, TAPR Regulatory Affairs Committee
43730 Vista Del Mar
Fremont, CA 94539-3204
dewayne@warpspeed.com

Abstract

This paper reviews the current status of Spread Spectrum (SS) in the Amateur Radio Service and also covers TAPR's activity on Spread Spectrum issues over the last two years.

Introduction

Back in 1989, Al Broscius N3FCT [1] discussed the use of commercially available Part 15 SS devices that were becoming available in the market, for use in the amateur radio service (ARS) for packet radio operations. He identified several commercial systems that were then available and made the following recommendation:

> "To responsibly address this technology, we feel amateur operators should experiment with the commercial systems now available in establishing long distance communication paths using high-gain antenna systems coupled with the maximum legal power of one watt, determining interference levels seen by weak signal receivers attributable to spread spectrum transmissions, and carefully introducing this technology to computer bulletin board operators who could financially support development of an unlicensed computer Internet."

To the author's knowledge, some effort has been made by the amateur radio community to pursue this recommendation. For instance, there have been reports by various hams of their experiences with such devices on various USENET newsgroups over the last several years; however, there has been little written about such experiences in ARS publications such as QEX, PSR or DCC proceedings. So while there are now millions of SS devices out in the world today in the hands of the average consumer, SS remains an unrealized technology in the world of amateur radio.

This paper will discuss some of the current activity that TAPR has undertaken over the previous year.

Amateur Radio SS Activities

Little has changed in the amateur radio service as far as high-speed Spread Spectrum (SS) packet radio is concerned in the period since 1989. Most of the commercial SS equipment available today on the market cannot be operated under the current Part 97 rules. One event of note however, was the publishing by the ARRL of the "Spread Spectrum Sourcebook" [2]. This was an excellent attempt by the League to acquaint the average ham with the technology of spread spectrum. Another excellent reference on packet radio technology and the use of SS appears in [3].

About the time of Apple's Data-PCS petition, Robert Buaas K6KGS submitted a request for an STA (Special Temporary Authorization) to amend Part 97 to allow relaxed usage of SS technology in the ARS. Buaas' request was granted by the FCC in 1992 and he was awarded an STA that had been renewed several times and remains in effect as of today. There is a recent QEX article which covers the STA and SS technology [4]. An earlier QEX article which describes the STA appears in [5].

Since the original STA was granted, Buaas and the other hams who are authorized to experiment under the STA have performed many experiments using SS technology both with existing Part 15 SS devices and homebrew hardware that was developed for the purpose of the experiments. This work formed the basis for the ARRL Board of Director's to pass a motion in January, 1994 to have their counsel submit to the FCC a petition for rule making to modify the SS rules in Part 97. Nothing was filed with the Commission in 1994, however the League's Board reaffirmed its decision in 1995 at their January meeting. In December of 1995, the ARRL filed for a rule making for the Amendment of Part 97 of the Commission's Rules Governing the Amateur Radio Service to Facilitate Spread Spectrum Communications [6]. The FCC assigned it as RM-8737 and comments and reply comments followed. Many of these are available on the TAPR SS web pages <http://www.tapr.org/ss>.

On April 10th, 1996, TAPR requested a waiver of the rules and regulations governing Amateur Radio spread spectrum communications in order to conduct an experimental program to test spread spectrum emissions over amateur radio facilities on different bands. Details on the request and subsequent STA see <http://www.tapr.org/ss/tapr_sta.html>.

In September 1996, TAPR released a position statement on Spread Spectrum Technology Development which in part states, "TAPR believes that the technical facts support our conviction that conventional and spread spectrum systems can coexist without detriment to conventional systems on all frequencies from MF to EHF. To this end, TAPR will begin to research spread spectrum systems that will develop technology for future deployment." (<http://www.tapr.org/ss/>) The full text is presented below.

On November 6, 1996 the FCC granted the TAPR STA request to conduct an experimental program to test Code Division Multiple Access spread spectrum emissions. By the end of January 1997, 61 TAPR members were participating in the TAPR STA on SS. At the end of 1996, TAPR was working with a manufacture of SS equipment that would

work under the STA. There was high hopes for this group purchase, but by the end of January '97, after 5 months of work, it became obvious that the Part 15 commercial manufacture had second thoughts and TAPR pulled the plug on the deal after the company continued to request agreements that could not be agreed to by TAPR.

On March 3rd, 1997, the FCC issued Docket 97-12 regarding a Notice of Proposed Rule Making with regards to RM-8737. The text is available at <http:/ /www.fcc.gov/Bureaus/Wireless/Notices/1997/fcc97010.txt>. The comments and reply comments that followed showed general agreement in some areas and disagreement in others. Much more work will be required in the future to get closure on this issue so that the ARS has rules that can best reflect what is required in the future for experimentation and technology adoption with the amateur radio service.

The next step will be for the FCC to issue a final rule making, but with all the recent staffing changes in the works at the FCC and the lack of consensus in the amateur radio community regarding the notice of proposed rule making, it might be some time before we see a final Report & Order.

On April 28, 1997, TAPR's initial six-month period of the TAPR STA ended. In accordance with the original terms of the STA, the TAPR program is on-going. Consequently, the applicants requested renewal of the TAPR STA, for an additional six months period. On May 6th, 1997, the STA extension was granted. In addition to submitting the request for renewal, the TAPR SS STA participants submitted an 80 page report on activity. This report can be found at: http://www.tapr.org/ss/tapr_sta.html.

TAPR's Statement on Spread Spectrum Technology Development

TAPR was founded in 1982 as a membership supported non-profit amateur radio research and development organization with specific interests in the areas of packet and digital communications. In the tradition of TAPR, the Board of Directors at their Fall 1995 meeting voted that the organization would begin to actively pursue the research and development of amateur radio spread spectrum digital communications. At the Spring 1996 board of directors meeting, the following statement of purpose was passed:

"TAPR believes that the technical facts support our conviction that conventional and spread spectrum systems can coexist without detriment to conventional systems on all frequencies from MF to EHF. To this end, TAPR will begin to research spread spectrum systems that will develop technology for future deployment."

As stated above, the TAPR board feels strongly about TAPR's focus on spread spectrum technology and especially how it relates to the potential coexistence on frequencies that will have increased number of users occupying them. The amateur radio bands, like other spectrum will become more heavily utilized in the future. It is in the interest of amateur radio to develop systems that are interference-resistant while not interfering with other primary or secondary users on those frequencies.

TAPR understands the concerns many have with the new technology, and believes that efforts in both education and research is necessary in order to allay the fears about interference and to demonstrate the benefits of the technology.

TAPR believes that todays' communications technology is moving toward all digital transmitters and receivers. These advances in technology, combined with the swift evolution of cell based transmission and switching protocols, are opening up a new set of possibilities for unique new services utilizing intelligent networks. These will contain smart transmitters, receivers, and switches. Today's Internet is perhaps the best example of a self-regulating structure that embodies these new technological approaches to communications in the networking domain. However, to date, many of these innovations have not moved into the wireless networking arena. TAPR will work on moving these innovations into the amateur radio community.

TAPR feels that the VHF/UHF/SHF radio networks of the future will involve a mixture of links and switches of different ownership, which terminate at the end-user via relatively short-distance links. What will then be required is a built-in, distributed, self-governing set of protocols to cause the network's behavior to make more efficient use of a limited, common shared resource, the radio spectrum. Creating such a self-regulating structure for the optimal sharing of spectrum will require much effort.

One of the major problems which stands in the way of these new approaches today is the current FCC regulatory environment and the manner in which spectrum is managed and allocated under its rules.

Historically, the current regulatory approach to radio has been based upon the technology that was in use at the time that the Communications Act of 1934 was framed, basically what we would call today, 'dumb' transmitters speaking to 'dumb' receivers. The technology of that time required reserved bandwidths to be set aside for each licensed service so that spectrum would be available when needed. Given this regulatory approach, many new applications cannot be accommodated since there is no available unallocated spectrum to 'park' new services. However, given the new set of tools available to the entrepreneur with the advent of digital technology, what once were 'dumb' transmitters and receivers can now be smart devices which are capable of exercising greater judgment in the effective use and sharing of spectrum. The more flexible the tools that we incorporate in these devices, the greater the number of uses that can be accommodated in a fixed, shared spectrum.

Therefore, TAPR will focus its spread spectrum effort in the following areas:

TAPR will work to promote rules and technologies to make the most efficient use of the spectrum through power control, forward error correction, and other means to minimize interference among spread spectrum users and existing communications systems.

TAPR will work on issues and efforts with other national organizations to change the regulatory environment and rules in order to promote the experimentation, development, and later deployment of spread spectrum technology.

TAPR will work to develop information on the topic to help educate members and the amateur community as a whole about spread spectrum technology, and to disseminate this information via printed publications, the World Wide Web, presentations at conferences and meetings, and other means.

TAPR will work to foster experimentation, development, and design of spread spectrum systems, and to facilitate the exchange of information between the researchers and other interested parties.

TAPR will work to develop a national intra-network to foster the deployment of future high-speed spread spectrum systems into regional and local communities, including the development of suitable protocols and guidelines for deployment of these systems.

TAPR will work with commercial companies who manufacture spread spectrum devices which operate in spectrum shared by the amateur radio service (ARS), in order to make them more aware of the nature of ARS operations on those bands with the goal to work towards the deployment of devices which will minimize interference between all spectrum sharing partners.

TAPR will work with commercial companies who manufacture spread spectrum devices in order to identify equipments that can be either used or modified for use for Part 97 operation.

Adopted by the TAPR Board on September 20th, 1996
at Seatac, Washington Board Meeting.

Spread Spectrum Statement Committee:
Greg Jones, WD5IVD
Dewayne Hendricks, WA8DZP
Barry McLarnon, VE3 JF
Steve Bible, N7HPR

The Future / Conclusion

TAPR plans to continue its leading role in developing standards and technology for spread spectrum communications for the amateur radio community through discussion groups, cooperative efforts and experimental programs such as now being permitted by the TAPR SS STA. In particular, due to the rapid development of communications hardware and software, TAPR believes that the use of hybrid spread spectrum emissions, as well as spreading codes not envisioned by Section 97.311(d) of the Rules can be employed without causing harmful interference to other amateur radio operations. Much of the debate over this issue has been voiced in the FCC SS rule making process and anyone interested should read through the comments and reply comments which are available on the TAPR website.

With the continued lack of available commercial equipment at a price that amateur radio operators can afford, the TAPR Board of Directors voted to fund the initial stages of a 900Mhz 200Kbps+ FHSS (Frequency Hopper) design. This is one of two designs that TAPR is undertaking in 1997. An initial design review of the radio should be published in the 1997 ARRL/TAPR DCC proceedings. The second design, which we believe will target either 2.4GHz or 1.2GHz and operate at speeds up to 1.544Mbps (T1), is on hold awaiting grant money to support the development project.

Like any volunteer effort, TAPR members and others interested in the technology will have to eventually become involved for the successful completion of many of these future projects. TAPR is always looking for members who want to be active in our Regulatory Affairs Committee or who want to participate in the TAPR SS STA. If you want to get involved it starts with asking and then doing.

References

[1] Broscius, A. "License-Free Spread Spectrum Packet Radio," 8th ARRL Computer Networking Conference Proceedings, p.16 [1989]

[2] American radio Relay League "Spread Spectrum Sourcebook," ARRL, [1991]

[3] Lynch, C.A. and Brownrigg, E.B. "Packet Radio Networks - Architectures, Protocols, technologies and Applications," Pergamon Press [1987]

[4] Price, H. "Digital Communications", QEX, p. 22 June, [1995]

[5] Rinaldo, I? "CDMA Spread Spectrum STA", QEX, p. 2 June, [1992]

[6] ARRL. "RM-8737" Available at: http://www.tapr.org/ss/arrl_filing_95.html [1996].

TCP Header Compression according to Van Jacobson via AX.25

Gunter Jost, DK7WJ/K7WJ
Lichtenbergstrasse 77, D-64289 Darmstadt, Germany
Translated by: Don Rotolo, N2IRZ
c/o RATS, PO Box 93, Park Ridge NJ 07656 USA

Abstract:

The Van Jacobsen scheme for TCP/IP header compression is briefly introduced, and an implementation of this system under FlexNet is described and discussed.

TCP/IP requires a header of at least 40 bytes for each packet. In connections over Ethernet carrying packets with little data, as in Telnet and similar mostly inactive sessions, this header adds considerable weight to be carried, making slow connections highly undesirable. Often there is only a little information carried within each packet, in the extreme case a single key click. Above all the reaction time becomes intermittently high. Once, when phone modems mainly worked at 2400 baud, Van Jacobson had taken up this problem and offered a technique in RFC 1144 [I].

With the Van Jacobson scheme, the header is compressed to be only 3 to 8 bytes in length. Interesting, and relevant to the implementation, is that the compression requires a memory table for each TCP connection. Only the differences from the last sent packet are transmitted. Frequently occurring special cases are handled in a specific manner and also improve the compression ratio. For example, the transmission of a TCP sequence number is mostly superfluous, because it can be calculated from the number and length of the packet.

TCP retries are generally carried uncompressed, so that the distant end can resynchronize. This is also valid for special, infrequent packet types (such as SYN and RST).

The necessity to store in memory the present status of each TCP connection requires naturally some computing capacity and memory space. The integration into the TCP layer was impossible, since it is a black box, and routers should also be able to handle compression. There is also the question of how many simultaneous connections should be realizable. For user stations 16 TCP connections should be sufficient, for routers it should possibly be more.

If there are more connections than slots available, the slot that is unused for the longest time is reassigned. It can also be a little tight with routers here. It is however not a problem to assign slots to stations using other criteria, for example according to the frequency of your usage.

The entire RFC won't be repeated here, mostly since it is over 40 pages long. The Protocol contains further details, and it is very interesting and well thought out. As far as the author knows, it has not been translated into German.

Implementation

The RFC contains already a practical implementation suggestion in C, which is a good platform upon which to build. Following a suggestion from Thomas Sailer, HB9JNX, I implemented the algorithm in the FlexNet IP coupler. To eliminate the need to implement the "hacks" Jacobson suggested for the transportation over SLIP, two new AX.25 PIDs were found to be needed for proper identification. After a review of all the available information, PIDs 6 and 7 were found to be available. When we informed the ARRL of this selection, they seemed surprised that we assumed them responsible for PID coordination, as was written in the AX.25 protocol definition some 13 years ago. At present, we are awaiting a confirmation of this coordination [2].

PID 6 denotes a compressed TCP/IP packet, with PID 7 an uncompressed packet being transmitted with the Jacobson scheme. That isn't the same as a packet without compression: In an uncompressed packet the slot-information remains, that is used for the compressed packets that follow. Naturally PID 8 used, as is known, for segmented IP packets. These also contain the original PID, and combinations of PID 6 through 8 as well as CC can appear. A segmented packet can contain a compressed header, thus it has PID 8 as well as the PID 6 or 7 packed within.

It is important to understand that the compression runs continuously point to point. The tables are organized as in an AX.25 QSO, and this functions only in Virtual Circuit mode. If a connection is rerouted or re-established, it must not be built on old status tables. One can not coordinate the compressiblity of an IP destination, as this belongs always to an AX.25 station or partner.

A disadvantage of the Von Jacobson specification is stated within, that the "partner" cannot provide the slot quantity required, but this should also not be different, since packets can get lost when the receiver has fewer slots than the sender. With the FlexNet implementation this is resolved by allocating a fixed 16 slots to the sender and up to 255 slots dynamically to the receiver, the maximum allowed under the specification. According to the definition, a Link Reset (reconnect) forces both sides of the connection to delete the relevant tables. In this manner, memory can be conserved.

There are no problems with data reliability. Each TCP packet is protected by a checksum. Additionally, all of the measures of an AX.25 connection are used. As long as the L2 connection remains, it is guaranteed that all packets arrive in a defined order. With a Link Reset, all the compression tables are deleted. In this way the defined continuation of the TCP connection is guaranteed, and the inadvertent confusion of a compressed packet from another TCP connection with the same slot number is safely eliminated.

Matthias Welwarshy, DG2FEF, has tested this behavior in WAMPES under LinuX, without the source code being exchanged between us. In this way, we have a high level of confidence that our implementations comply with Jacobson specification. In a multi-week test, run between four stations at different user ports, there were no unmotivated disconnections of the TCP connections. Last but not

least. this shows that the behavior is very robust against sporadic re-establishments of the 1ogical (AX.25) connection.

Conclusion

We can conclude that the Van Jacobson compression scheme provides a very good result with minimal difficulties. One must ask himself, why this concept has not (in the author's knowledge) been previously applied to Amateur Radio.

One disadvantage is that, at present, one of the two partners must switch the compression on manually. It must be assured that the other partner controls the system. In a further development, this facility should be handled automatically.

References:

[1] Jacobson, V., Network Working Group, Request for Comments 1144, February 1990.
[2] Also awaited for quite some time is the coordination of PID CE for FlexNet.

TCP/IP on FlexNet - Just Another Layer

Gunter Jost, DK7WJ/K7WJ
Lichtenbergstrasse 77,D-64289 Darmstadt, Germany
Translated by: Don Rotolo, N2IRZ
c/o RA TS, PO Box 93, Park Ridge NJ 07656 USA

Abstract:

The goals and outcome of a project to optimize TCP/IP transport over the FlexNet AX.25 network is described. A number of optimizations, and their implementations, are described and discussed. These include header compression, resend minimization, packet age tracking and ACK consolidation, as well as platform considerations and potential uses.

Not long ago, TCP/IP was an exotic mode of operation in Amateur Radio, reserved for specialists. Since then, it has become one of the most popular protocols used in the digital modes, due to the wealth of interesting applications employing it. The project described here attempts to make TCP/IP usage in the FlexNet packet network as simple as possible.

This doesn't mean that we can dispose of the de-facto AX.25 standard. No, TCP/IP is merely a useful expansion, or just another layer, that can be handled by FlexNet.

This also includes the reconnection of disconnected links without the intervention of the operator, including re-routing to faster links. TCP/IP can be seen as an end-to-end layer (Layer 4 in IS0 jargon), in contrast to other packet systems that are hop-to-hop layers, namely to the real ends of a logical circuit (user/user or user/server). TCP/IP strengthens the robustness of the AX.25 connection to the entry node, allowing one to change user ports during a running connection, without which the connection must be newly established. Instead of creating our own Layer 4 specification (plans have existed for some time), we find that TCP/IP offers exactly that, and is available for most modern operating systems. TCP/IP is the standard protocol of the Internet, and there is a considerable amount of software available.

TCP/IP and FlexNet make up a homogeneous unit, in that FlexNet nodes need no further expansions or changes, because they are already fully transparent to all frame types offered by the protocol, which are passed on a virtual AX.25 L2 connection. The protocols are passed along cleanly, as long as AX.25 (on the RF path) and TCP/IP (end-to-end) run at the layers they are expected to be at.

In our first look it was quickly seen that TCP/IP, as presently used in Amateur Radio, was given the (not undeserved) reputation of being slow and inefficient. This isn't the fault of the protocol, but of the implementations. It was clear where the crank needed to be turned. Quite simply, it shouldn't be like that, so we began this project. Today, a few thousand lines of source code that were implemented and tested on the first platform (Windows 95) are ready for use.

The idea of placing TCP/IP services directly on the network as AX.25 shouldn't be ignored. This is an interesting concept, but it should be pointed out that a careful TCP/IP implementation will return more than it costs.

Minimizing Protocol Overhead

TCP/IP packets contain a header of approx. 40 bytes. For a 256 byte packet, this is an overhead of some 16%, and, when the required ACKs are considered, more than 30% overhead increase as compared to an AX.25 connection. This also assumes an optimal protocol implementation, as well as no unnecessary retries. If you lengthen the packets to 1500 bytes (a typical value on Ethernet and similar implementations), the overhead sinks to a more reasonable 5%. A lengthening of AX.25 frames to 1500 bytes, as is often suggested, isn't practical. The frame failure rate would become unreasonably large: If an RF link of 256 byte frames is 90% failure free (a realistic (unfortunately) number), this value would decrease to only 40% error-free frames for 1500 byte packets. Sporadic interference such as radar impulses further worsen this value. For this reason, Ax.25 segmentation was defined some time ago, so that a large IP packet could be broken into many 256-byte packets. In this way, each frame contains only its own additional overhead bytes, contributing to efficiency. A requirement for this to function is that each packet arrives in the correct order, which can only be safely realized through a VC (Virtual Circuit) connection. The FlexNet AX.25 header compression feature *[implemented some time ago - IRZ]* functions in any case only with such a static connection and is a source of further reduced overhead.

These improvements are most efficient when there is a relatively large amount of data to be sent (enough to permit segmentation). If this isn't the case, such as in an interactive Telnet session, the value becomes somewhat less. The worst case is when each byte requires its own packet, with its 40 byte overhead. In these cases one needs some other mechanism. Luckily, there is a standardized header compression scheme, as seen in RFC 1144 [1], and implementation was relatively simple. Using the Van Jacobson scheme [2], the 40-byte header of a static TCP connection can be reduced to 3-8 bytes. the only assumption for this to work is, like the FlexNet header compression scheme, the connection path between both ends is static, i.e. a VC. This protocol was defined for relatively slow telephone connections. If you replace the concept of "Serial Line" with "AX.25 virtual circuit", you can see just how well the protocol would fit with AX.25 encapsulation. The compression and its control occurs on the AX.25 level for up to 250 TCP connections, as well as traffic forwarded through a router. With a Link Reset, the status tables on both sides of the path are re-initialized. This requires that both ends of the path know of these Link Resets, but not all AX.25 implementations do this, which can lead to problems in this area.

Thinking instead of Pumping

A general problem with end-to-end protocols is that the transport shell has only limited possibilities for influence. This is less important for a static TCP connection, where the timers regulate the connection fairly well on the basis of the transport capacity, but nonetheless, an orgy of unnecessary retries can occasionally be observed.

The goal of this project was to develop an IP router not only for usage under Windows 95, but for usage under DOS or on the RMNC platform. This gave us a reason to invest a little more work early on, to save work in the later cross-platform portings. Finally, we still have to co-exist with users and servers operating with network implementations that are sub-optimal.

With AX.25 these problems, once separated from channel control, are mostly resolved. The network nodes decouple both sides of a connection completely, and traditional digipeating is no longer practiced. With this philosophy, to an end-to-end approach such as IP becomes an attractive proposition.

IP is a connectionless protocol, and the TCP placed upon it is laid out such that packets can take any possible path from one end to the other, arriving in any order. For this reason, it is possible to route individual IP packets in a connectionless manner. An IP router might never see all of the packets of a particular connection. It is also true that when you encapsulate something within AX.25, it cannot be assumed that traffic from one side of a connection travels over the same path as traffic from the other side. Only at the endpoints can you be sure of seeing all packets of a TCP connection.

An IP router also has possibilities for optimization. If a router knows of congestion, for example the outgoing path in use is slower than the incoming path (not an unusual condition in a Packet network), it can analyze packets in transit stored in memory and eliminate all unnecessary resends, by simply erasing any doubled packets. A time stamp can also be put onto each packet. Packets that remain in the router for some time, say a minute (because of congestion, or a broken link) can be erased, hopefully resulting in a TCP retry from the sending station that might be routed along a better path. Implementation of this requires an analysis of the complete IP and TCP headers, so that the order of packets can be determined. This can also be used to consolidate ACKs for a specific TCP connection, passing only the latest to the endpoint.

Through such actions, congestion created by unnecessary resends can be dealt with instead of becoming worse. With traditional implementations, traffic is brought to a complete halt in such situations. Through controlled 'packet loss', as well as the deletion of doubled or aged packets, the net is much better able to deal with changes, slowing the data to match the path's capacity.

This behavior functions very effectively and completes the usual improvements, with the router sending an ICMP-source-quench as well as requesting the sender to throttle back. The retention of the originator is during this not clearly specified, thus such flow control actions can lead to very different results.

Naturally, both partners should adjust their TCP timers properly, but luckily this functions quite well, even with the Microsoft stacks.

The proper adjustment of TCPIIP parameters is critical, above all when one wants to reach destinations via different networks. The Microsoft stack is a "black box", and hardly any parameters can be adjusted. So, we need to make all optimizations in the module which knows the existing transport layer the best: The AX.25 coupler.

Instead of simply pumping TCP/IP packets into an AX.25 connection or, even worse, into AX.25 UI frames, as is done with present implementations (such as LinuX), the packets remain in the IP coupler's (e.g., router's) memory until they can actually be sent. This is comparable to the behavior of the FlexNet coupling to Layer 1, and contrasts with solutions such as KISS, which only generates frames that can be sent immediately. The known side effects of a KISS connection, i.e. multiple SABMs or RRs in a single transmit cycle, simply don't occur with FlexNet.

In FlexNet, ACKs are sent only for the latest packet, any ACKs belonging to earlier packets need not (and are not) sent. A carryover of this concept to the IP-AX.25 coupling brings some improvements. When the AX.25 connection is being established, during which no data can be passed, running TCP/IP packets are buffered, and so can cause unnecessary retires. The same happens with a user station that is blocked in a half-duplex connection for extended periods. During this time is it possible for many TCP packets to pass to the user station, each of which is ACK'd individually by the user's local TCP stack. Instead of sending each of these ACKs, the FlexNet coupler can discard all of them except for the last one. Implementing this requires a close coupling of the layers, but FlexNet already has the call in it's API, so that little accommodation is needed here. Naturally, the coupler must analyze and compare the packets at the TCP level, to be able to discard the proper packets. In the ideal case, a user station sends a maximum of one AX.25 I-frame during its time slot per active TCP connection, instead of the many L2 RRs and TCP ACKs as with the traditional solutions. The improvements thus realized are somewhat greater than those obtained from header reduction alone. While header reduction remains important, one cannot ignore the gains in efficiency from eliminating doubled or aged TCP packets.

In a typical HTTP session, many TCP connects are started, transferring only a few kilobytes of data before being closed again. TCP timers have no chance to adjust, and many unnecessary retires occur. This is especially true in a slow half-duplex environment which is typical for a user station. The situation is less dramatic when users have few, long-lasting connections, e.g. a single FTP transfer.

For AX.25, FlexNet was able to make such optimizations directly in the L2 code, because of the tight coupling to Ll . This avoids the need for the L1 process to have to analyze each L2 frame and decide what should be passed ahead or not. Unfortunately, it isn't so easy on the TCP level. Here, you must watch the status of possibly many running TCP connections, which requires a lot of memory and efficient algorithms. However, some coding efficiencies result because TCP compression needs quite similar data structures, which can be partially be used for these optimizations.

All this naturally increases demands upon memory space and CPU capacity. As long as the network runs at data rates below 100 kB/s, and PCs continue to increase in capacity, no further effort is needed. This is especially true on the user side, where nowadays there is more than enough computing capacity. A fast 486-class machine with 16 MB of RAM running Windows 95 is sufficient to support a throughput of 76,800 baud TCP/IP through FlexNet. For a router it would help to install a faster CPU, but reasonable performance with a 486 was measured.

Anachronisms

All the improvements described above assume IP transport over AX.25 Virtual Circuits, which should provide a reliable path between two points of an IP network. Using Datagram mode (AX.25 UI frames), each packet loss on the wireless portion of a TCP connection causes a TCP retry, which has to travel the whole way, end to end. (Remember the problems we had with L2 digipeating? Same applies here). Since this has not been generally learned and understood, an IP router must also be able to deal with the Datagram mode, where IP packets are sent as AX.25 UI frames. With this, one becomes stuck on the problem that only IP packets with a gross length of 256 bytes or less can be carried. AX.25 segmentation is not usable, as it depends upon the packets traversing the network in their original order.
To resolve this problem, we have available so-called IP fragmentation, which divides an IP packet into multiple smaller packets and gives (in contrast to AX.25 segmentation) each fragment it's own complete

IP header. Each packet is thus autonomous, and are reassembled in order only at the destination. This allows each packet to traverse the network through any available path, forbidding reassembly in routers or gateways. One problem is that when one fragment is lost, the entire packet (a series of fragments) must be re-sent. This scheme is clearly at best a temporary patch, especially when one must deal with the realities of the RF medium and the relatively high failure rate of the links. For compatibility reasons, Datagram mode is presently the only way to deliver a fragmented packet, and thus IP fragmentation must be implemented in the router.

IP Routing

IP routing is, in one regard, in competition with established AX.25 routing, especially in the FlexNet environment. In that IP addresses are mated to a callsign (dynamic address allocation for users is a subject for separate discussion), FlexNet routing is sufficient to reach a specified destination. One useful addition could be to define a layered hierarchical routing, such as AX.25 routing in the local area and IP routing for the wide area, analogous to protocol layering. An obstacle to this is that our network is not yet fully developed to offer fast connections with transit times of less than a minute over its entire range.

One argument in favor of IP routers at network facilities is that the user is freed from some work - he simply sends packets to the router, and it handles the rest. At the moment this merely means that the sysop's work, in which he might not have much interest, is delegated. The user shouldn't become dependent upon this system, however, he must retain the possibility of establishing a connection with a destination directly. This means that PID transparency, reversible callsign fields and a functional AX.25 router remain important. The network can be considered a cloud, its internal operation the user need not know to reach a destination.

With more and more users operating TCP/IP (perhaps in part due to this project), and TCP/IP is understood to be the end-to-end layer on a well-constructed AX.25 network, we can then reconsider the optimal bundling of traffic via dedicated routers. The infrastructure is, in the form of many LinuX systems, more and more available. All that was missing were the protocols and their implementation.

If the TCP traffic is optimized as discussed above, then routers can reduce network loading considerably. Routing, however, remains in the background. In any case, this is an exciting direction for development and there is considerable room for new ideas.

Until then, and possibly also after that, the user is better served with making a direct connection himself, as far as the network allows.

Which platform is the right one?

While we lean towards additions to some specific operating system, the rest of the world wants missionaries. Indisputably LinuX, for example, is an outstanding server/operating system. The average user, however, wants to simply buy something from Microsoft. You just can't beat Windows 95 or NT when you want to develop a user application.

For Windows 3.x, ETHEREMU from Thomas Sailer, HB9JNX, could be installed. It emulates an ethernet card and allows it to communicate via Trumpet Winsockets. The AX.25 routes are input using text mode commands.

TCP/IP is already integrated with Windows 95. Of course data transfer via Ethernet or "DFU-Network", using SLIP or PPP protocol, is provided. Until now, what was missing was a coupler that could encapsulate TCP/IP packets within AX.25. Some solutions already exist, but are generally difficult to configure, for example NOS in a DOS-box. Others try to use protocols such as SLIP, which has the disadvantage of removing the possibility of being able to have normal AX.25 connections. Besides, one needs special, and not quite inexpensive, hardware.

In the meantime, PC/FlexNet runs under Windows 95, all that's missing is the coupler that hands the IP packets over to FlexNet. Microsoft makes this possible only via the installation of a "Network Card", however the concept is, at least in the German translation, somewhat misleading. Of course, all of the optimizations described have been implemented, using Windows 95 as a test platform. Although the timing of the TCP stacks is really set up for a fast wire connection, it keeps stations on the RF channel fairly civilized and sends practically no unnecessary packets. A list box is used for AX.25 route inputs, and it also serves as a status and statistics display. No special TNC hardware is required, and running multiple IP sessions in parallel with AX.25 sessions presents no problems. However, this is a solution mainly for users, i.e. network clients.

As a server system, Windows 95 is somewhat strained. While there are some simple FTP and HTTP servers that function well, the entire system in not stable enough for use as a general-purpose server. One negative is that there is (still) no possibility of forwarding IP between multiple ports, as well as AX.25 to ethernet ports. This is the domain of LinuX, but NT is close behind. LinuX is already available with AX.25, so the next development point lies with NT. Having a large installed user base for 95 and NT doesn't hurt, either.

Uses

As already mentioned, applications and services are sufficiently available. Much that is developed for the Internet is also usable in Amateur Radio. Regardless, that shouldn't prevent someone from developing an amateur-specific program. Services such as databases or special applications like DX-Clusters are easier to access and service using TCP/IP instead of AX.25. Thanks to TCP port addressing, it's no problem to offer and try various applications and services on the same server with the same AX.25 callsign.

A further field for new applications is for image and speech transfer. While these demands today seem somewhat utopian, voice mailboxes are already using the network to transfer their messages via store & forward. It isn't a much larger step to make it possible for the user to have a direct digital connection into the system. The software already exists, though this is mostly designed for the relatively high speed of the Internet. If you don't need real-time and full duplex, and can deal with PTT (amateurs know this already) one can get very good results. Modern codecs allow speech to be compressed to less than 300 bytes/sec with little loss in quality. This bandwidth is already available in most of the network. In the future, a Windows application will be introduced. With specific servers it should also be possible to have roundtable discussions like on the local FM repeater.

Image transmission also remains in the range of possibility, especially with the quality codecs available for moving picture transmission. Naturally the existing available bandwidth won't work for decent quality video, but you don't really need it to admire someone's photograph. Convenient video conference systems are realizable in the foreseeable future.

All this carries with it the ability to increase the play value of the network, and through that, justify the spectrum being used. Naturally, our packet-oriented data transfer system isn't set up for real-time uses. The protocols and applications to be used require careful consideration of our unique requirements and conditions, but there is a lot of room for experimentation here.

Unquestionably, a careful optimization of all parts of the transfer chain is needed. What is lost in one component, whether in hardware or in software, cannot be recovered. For example, while increasing the baud rate is a good idea, it is not the only possibility for improving the network. Clearly, TCP/IP can bring to Amateur Radio much more than just a wireless Internet.

References:

[1] Jacobson, V., Network Working Group, Request for Comments 1144, February 1990.
[2] See also "TCP header compression according to Van Jacobson via AX.25" (Jost) elsewhere in these proceedings.

An Amateur 900 MHz Spread-Spectrum Radio Design

Tom McDermott, N5EG, n5eg@tapr.org,
Bob Stricklin, N5BRG, n5brg@tapr.org,
Bill Reed, WDOETZ, wdOetz@tapr.org

Abstract

System design principles and high-level design details are described for a new spread-spectrum radio design for the 900 MHz. Amateur band. The radio is designed to provide a 10-base-T interface as the data port, and is designed to provide transport of IP-based data. It is planned to provide both stand-alone and fully-networked hub configurations. The design is based on Frequency-Hopped Spread Spectrum (FHSS) spreading. Use of Forward Error Correction (FEC) and QPSK modulation should provide significant system gain performance compared to other FHSS FSK designs. The radio is currently in the printed-circuit board layout stage.

Introduction

Significant enhancement in the use and application of computer networking in the last 5 years has led to the need for high performance wireless interconnection of computers. Traditional 1200-baud and 9600-baud packet links are not able to provide adequate speed for today's web-based applications. Further, long-haul linking of multiple radios in linked configurations has proven difficult and unreliable. This can be seen from simple numerical analysis of the poor reliability of such multiple-hop configurations'. One solution to the reliability issue is to utilize other transport facilities for most of the transmission distance, such as the Internet.

In industry, wireless is valued greatly for the ability to provide mobility. Thus, fiber optics has replaced radio in the long-haul telephony networks (for most, but not all applications), and wireless is increasingly looked upon as a replacement for the wire copper loop. This inverts the traditional view of the wired and wireless domains*.

Applications

A high-speed mobile data access infrastructure to the Internet has many applications for the radio amateur, and could allow the provision of services and applications not possible with current commercial technologies. This is especially true as the Internet performance improves to support constant-bit rate multimedia services. Current audio coding technology provides quite acceptable audio at 13 kb/s. Videoconferencing is reasonably acceptable at 112 kb/s. Web browsing is possible at any speed, but only tolerable above 28 kb/s. A wireless interconnection technology that could support data rates in this range would provide the ability for the radio amateur to provide audio conferencing, via the Internet, from a mobile laptop computer to anywhere in the world in real time. Mobile laptop videoconferencing is similarly

possible. Access to databases, maps, Email, etc., anywhere on the Internet in real time would make the utility of such a service very great. The radio amateur, equipped with such a capability could prove invaluable in many public-service scenarios. Indeed, the Internet not only addresses many of the problems of previous-generation packet networking, in fact it provides a powerful tool in its almost universal accessibility and rich diversity of information.

System Requirements

The design of a radio to meet the above applications is described. The general requirements are that the radio provide at least 128 kb/s throughput (more in other modes) while providing 20-mile coverage with l-watt output power. 1 O-base-T was selected as the desired interface, and it is intended for connection to the LAN port of a laptop or other computer. It is envisaged that both a point-to-point configuration and a hubbed multi-point configuration would be supported. In the point-to-point configuration the radios would simply provide a transparent LAN interconnection pipe. For example, one radio might be connected to an Internet service, and located on top of a tall building, while the other end would be connected to a mobile laptop computer.

In the multi-point configuration, several radios are placed at a common site, such as a tall building. One channel becomes the control channel, and each of the remaining radios serves as a data channel. This provides for multiple users to simultaneously access the hub site. In the hub mode, all radios transmit and receive in synchronism. Additionally, good Internet connectivity might not be available at such a hub site, so individual data channels of the hub can be dedicated as fixed point-to-point links that provide a remote link to the I nternet from the hub site. The radio design supports these configurations automatically with additional hardware. The control channel allocates access to idle data channels.

In the hub mode, the hub provides for dynamic assignment of IP addresses to the user computers via the DHCP protocol. This eliminates many of the difficulties of IP address administration in a mobile environment. However, it does not allow the user to move the computer from one node to another while connected. Instead the link will be broken and will have to be re-established with a new IP address.

Spreading Methods

Both Direct-Sequence Spread Spectrum (DSSS) and FHSS were studied. The Harris Prism™ chipset was initially investigated for such a radio. This chipset is designed to provide 802.11 wireless LAN for mobile laptop computers. However, this excellent chipset cannot easily provide the required system gain and performance required for a 20-mile link. It was intended to provide a low-cost low-power 1 Mb/s LAN interconnection primarily within a few 100's of feet. The Prism chipset utilizes DSSS modulation, and provides a spreading gain of only 12 dB. maximum, 11 dB typically. Further it is designed for the 2.4 GHz. band, which we felt would be difficult for average amateurs to equip with adequate antennas and feedline to meet

the link distance requirement. We chose to implement the first radio design in the 900 MHz. Amateur band (902-928 MHz., a width of 26 MHz.) due to the availability of commercial components.

At first blush 20 dB of system gain (100:1 spreading ratio) within a 26 MHz wide band implies a maximum data rate of 26/(100*2) = 130,000 b/s. Since we also wanted the radios to operate half-duplex (to minimize cost), this maximum rate would be further reduced to 65,000 b/s. The data rate could be doubled if QPSK modulation is utilized, because it halves the spectral requirements. However, we noted in several spectrum analyzer sweeps of the 900 MHz band in Dallas, Texas that a large number of very strong narrow-band carriers are present. Testing with commercial part-15 radios indicated that these strong carriers render DSSS radios inoperative when the link distance was increased beyond one or two miles.

However, tests with FHSS radios under the same conditions proved to be more encouraging. Eventually, 20-mile links were achieved with one FHSS radio when the antennas were converted to horizontal polarization. Horizontal polarization reduced the amplitude of the interfering carriers by more than 20 dB. Thus an FHSS-based radio design was selected.

System Design Parameters

The parameters that were initially selected for the radio design are based on the availability of off-the-shelf SAW filters for the IF strip, what we felt was an achievable settling time for the frequency hopping VCO, available integrated circuits, and an aggressive but hopefully reasonable demodulator synchronization time. These parameters have been selected as follows:

Dwell time on each slot:	10 milliseconds
IF filter bandwidth:	**600** kHz.
RF instantaneous bandwidth:	**600** kHz.
RF channel bandwidth:	**26** MHz.
Number of slots within band:	**43**
Modulation format:	QPSK, square-root raised-cosine roll-off
Forward Error Correction:	Convolutional, based on K=7 coder and Viterbi decoder. Code rate = 1/2 or 7/8 depending on mode.
Frame structure:	Based on HDLC frame
Demodulator:	Digital Costas-loop design
Modulation rate (all modes):	300 kilo-symbols/second
Transmit / Receive mode:	Time-Division Half-Duplex
Data throughput (mode 0):	150 kb/s (minus overhead)
Data throughput (mode 1):	300 kb/s (minus overhead)
Data throughput (mode 2):	-525 kb/s (minus overhead)

Table 1 indicates the modes of operation that are anticipated.

The use of FEC and QPSK provides at least 9 dB improvement in system gain as compared to uncoded non-orthogonal Frequency-Shift Keying (FSK) which is utilized in almost all commercial part-l 5 radios. However, the use of coherent modulation techniques increases both the cost of the radio and the difficulty of the design. We felt the 9 dB. performance improvement made this tradeoff worthwhile. Fortunately, Harris provides a DSP-based digital Costas-loop QPSK demodulator IC (the HSP 50210) which appears to have sufficient programmability to meet the synchronization speeds provided that some clever algorithms ("quick-lock") are employed.

Two risks are felt to represent the greatest challenges in the radio design. First is the ability of the hopping VCOS to settle to adequate frequency accuracy and stability within 10 milliseconds. Second is the ability of the Digital QPSK loop demodulator to achieve synchronization lock with our special "quick-lock" technique. The prototype design will be used to assess these design risks.

Block Diagram

Figure 1 is a block diagram of the baseband processing, processor, and LAN Interface portions of the radio. Figure 2 is a block diagram of the RF and IF processing parts of the radio. The radio design is based on a Motorola 68360 microprocessor. It controls all major functions of the radio, and the LAN interface. A Motorola 68160 provides the 1 O-base-T Ethernet port. FLASH memory is utilized solely in the processor, to allow updates of the code at a later time without physically opening the radio or removing / programming any EPROMS.

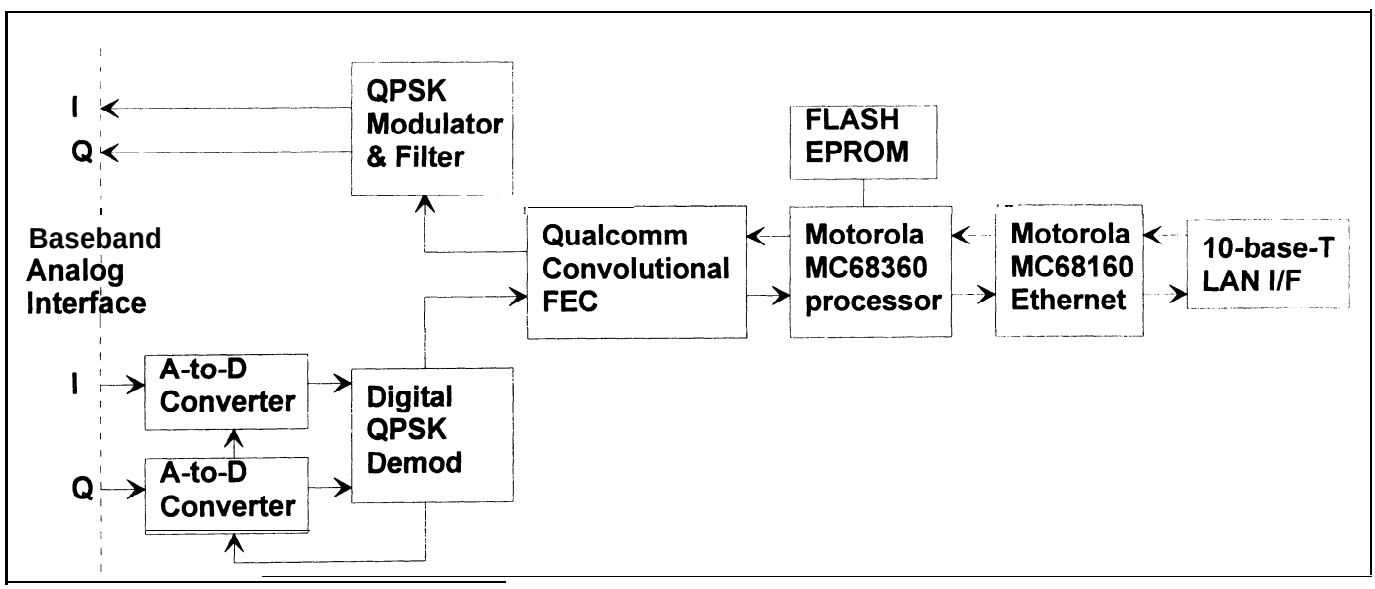


Figure 1 – Block Diagram: Baseband Processing and LAN Interface

Circuit Description - Transmit Direction

The data from the LAN port is buffered by the 68360 and converted to a proprietary frame format based on HDLC and then sent to a Qualcomm convolutional coder IC. In modes 0 and 1, the coder produces two output bits for each input bit (rate = 1/2 mode). In mode 2, the code is punctured to rate = 7/8. These two bits become the in-phase (I-) and quadrature (Q-) channels to a Motorola QPSK modulator IC. The modulator IC provides raised-cosine roll-off at baseband of the two channels via an FIR filter. It also contains two D-to-A converters, and thus provides the I- and Q- analog baseband output signals.

The two baseband analog signal are connected to a Harris quadrature up-converter IC that

generates I- and Q- signals at the IF frequency of 85.35 MHz. These signals are then further upconverted to the 902 MHz band, and filtered by a dielectric filter to eliminate the IF image frequency. It is then amplified by a Motorola integrated PA chip to about 100 milliwatts. The signal is routed through a PIN diode switch and through a pair of directional couplers to the antenna connector. The directional coupler signal are rectified and filtered, and fed to an A-to-D converter chip. These signals provide measurement of the forward and reflected power levels.

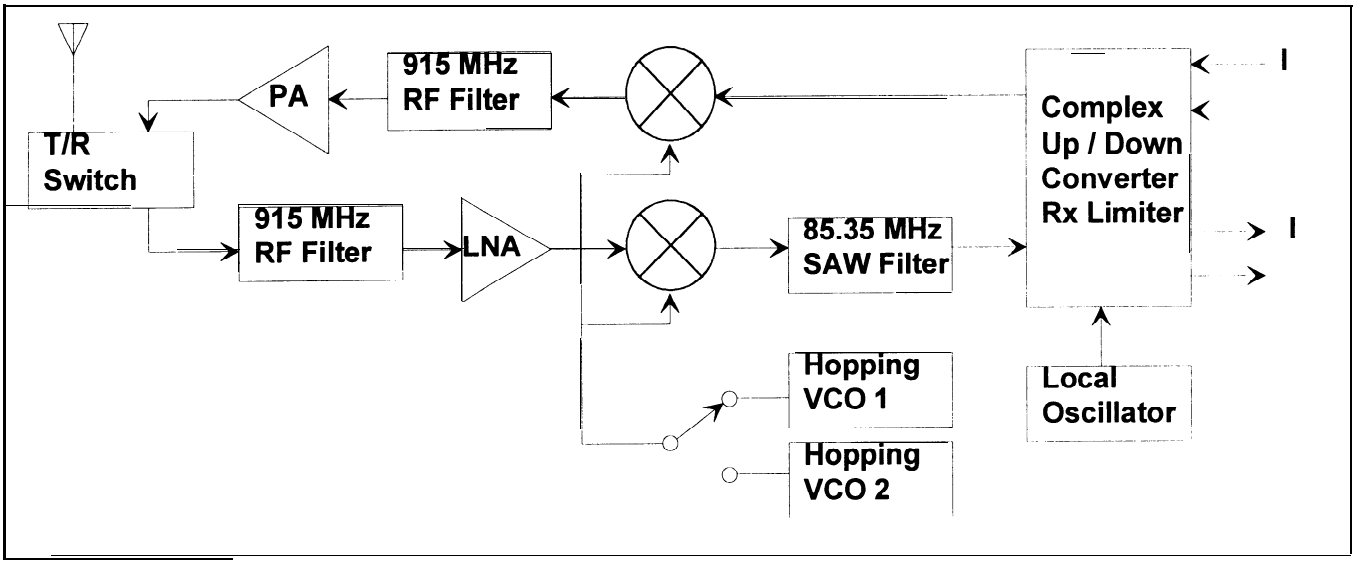


Figure 2 – Block Diagram: RF and IF processing

Circuit Description - Receive Direction

In the receive direction, the signals are passed through a dielectric filter (to eliminate the image frequency) and then to a Motorola low-noise downconverter IC. From there they pass through an 85.5 MHz, 600 kHz wide SAW filter and an amplifier. At that point, they are sent to a Harris downconverter IC which provides a large amount of gain through a two-stage limiter, and then downconverts the signal to baseband, producing the I- and Q- baseband analog signals. These signals are then digitized by a pair of IO-bit A-to-D converters, and sent to the Harris digital Costas-loop demodulator IC.

The demodulator IC first performs a complex frequency rotation to adjust for any frequency offset and phase error between the transmitter and receiver, then provides symbol timing and carrier frequency acquisition and tracking. Finally it provides AGC on the demodulated baseband signals, and performs a soft-decision threshold comparison of the I- and Q-channels against the reference level. These are in the form of two 3-bit words, one for the I-channel, and one for the Q-channel.

The pair of I- and Q- soft decision signals are sent to the Qualcomm Viterbi decoder IC. Is it capable of determining the synchronization boundary of the QPSK symbols, and decoding the FEC algorithm. The decoded bits (at one half the rate of the input bits in modes 0 and 1) are then sent to the HDLC portion of the Motorola 68360. The microprocessor recovers and removes the HDLC frame, and transmits the received data out the IO-base-T LAN port via the 68160.

Hopping VCOS

The design utilizes two VCOS in a pair of phase-locked loops (PLLS). While one loop is operational on frequency, the other loop is busy slewing to a new frequency. At the end of each IO-millisecond period, the new VCO becomes the active VCO and the previously active VCO is slewed to another channel. In this manner, each VCO plays leapfrog, being utilized half the time. This allows each phase locked loop 10 milliseconds to achieve satisfactory frequency accuracy before it is switched into service.

All of the RF-determining reference frequencies are derived from a single crystal-controlled oscillator. This oscillator is ovenized to minimize its error from the desired frequency during temperature excursions.

The actual programming of the VCO PLLS occurs by a small PIC chip (one-time programmable single chip processor). This chip contains the hopping sequence of the radios, and cannot be altered by the user. United States Department of Commerce regulations prohibit the export of FHSS radios from the United States if the hopping sequence can be altered by the user.

Synchronization

The most difficult part of any design is the synchronization of the transmitter and receiver, both in terms of the Transmit / Receive switching (T/R) and also in terms of carrier frequency acquisition. An initial synchronization interval occurs prior to the radios becoming linked. This takes some time to occur. The demodulator utilizes a sweeping process to recover carrier lock. However once this is achieved, the microprocessor is capable of reading out the frequency error at the receiver demodulator from the acquisition register in the demodulator. Based on the actual RF channel utilized during the initial synchronization, it computes the master-oscillator frequency difference between the transmitter and receiver. Subsequently, each time that the radio hops channels, the microprocessor computes the new effective

frequency difference, and pre-loads the demodulator carrier recovery loop register with the proper frequency offset value to place the recovered carrier very close to the proper frequency. This helps the demodulator lock very quickly. This is the “quick-lock” technique referred to earlier.

Acknowledgements

We would like to thank the Tucson Amateur Packet Radio Corporation (TAPR), which is sponsoring this project.

Table 1 – Proposed Operational Modes

Mode	End Points	Performance	Throughput
PPS	Point-to-point search	Search mode to establish initial link in PP mode.	
PP0	Point-to-point (i.e. user end system to user end system)	Rate=1/2, half-duplex, 10 msec T then 10 msec R.	150 kb/s
PP1	Point-to-point (i.e. user end system to user end system)	Rate=1/2, transmit slots as needed, communication of slot requests across link.	300 kb/s
PP2	Point-to-point (i.e. user end system to user end system)	Rate=7/8, transmit slots as needed, communication of slot requests across link.	525 kb/s
PNS	Point-to-Control Link (i.e. user end system to control link of a multi-radio node)	Search mode to establish initial link to control channel of a node.	
PN0	Point-to-Node (i.e. user end system to data channel of a node).	Rate=1/2, half-duplex, 10 msec T then 10 msec R.	150 kb/s
PN1	Point-to-Node (i.e. user end system to data channel of a node).	Rate=1/2, transmit slots as needed, communication of slot requests across link, with node doing slot voting across all channels and downstream notification to all users.	300 kb/s
PN2	Point-to-Node (i.e. user end system to data channel of a node).	Rate=7/8, transmit slots as needed, communication of slot requests across link, with node doing slot voting across all channels and downstream notification to all users.	525 kb/s

[1] A Primer on Reliability as Applied to Amateur Radio Packet Networks, T.C. McDermott, N5EG, 1 3th ARRL Digital Communications Conference proceedings, pp. 122-l 25

[2] This effect has sometimes been called the “Negroponte Inversion”, after Nicholas Negroponte.

VHF/UHF/Microwave Radio Propagation: A Primer for Digital Experimenters

Barry McLarnon, VE3JF
2696 Regina St.
Ottawa, ON K2B 6Yl
ve3jf@tapr.org

Abstract

This paper attempts to provide some insight into the nature of radio propagation in that part of the spectrum (upper VHF to microwave) used by experimenters for high-speed digital transmission. It begins with the basics of free space path loss calculations, and then considers the effects of refraction, diffraction and reflections on the path loss of Line of Sight (LOS) links. The nature of non-LOS radio links is then examined, and propagation effects other than path loss which are important in digital transmission are also described.

Introduction

The nature of packet radio is changing. As access to the Internet becomes cheaper and faster, and the applications offered on the "net" more and more enticing, interest in the amateur packet radio network which grew up in the 1980s steadily wanes. To be sure, there are still pockets of interest in some places, particularly where some infrastructure to support speeds of 9600 bps or more has been built up, but this has not reversed the trend of declining interest and participation. Nevertheless, there is still lots of interest in packet radio out there - it is simply becoming re-focused in different areas. Some applications which do not require high speed, and can take advantage of the mobility that packet radio can provide, have found a secure niche - APRS is a good example. Interest is also high in high-speed wireless transmission which can match, or preferably exceed, landline modem rates. With a wireless link, you can have a 24-hour network connection without the need for a dedicated line, and you may also have the possibility of portable or mobile operation. Until recently, most people have considered it to be just too difficult to do high-speed digital. For example, the WA4DSY 56 Kbps RF modem has been available for ten years now, and yet only a few hundred people at most have put one on the air. With the new version of the modem introduced last year, 56 Kbps packet radio has edged closer to plug 'n play, but in the meantime, landline modem data rates have moved into the same territory. What has really sparked interest in high-speed packet radio lately is not the amateur packet equipment, but the boom in spread spectrum (SS) wireless LAN (WLAN) hardware which can be used without a licence in some of the ISM bands. The new WLAN units are typically integrated radio/modem/computer interfaces which mimic either ethernet interfaces or landline modems, and are just as easy to set up. Many of them offer speeds which landline modem users can only dream of. TAPR and others are working on bringing this type of SS technology into the amateur service, where it can be used on different bands, and without the Effective Radiated Power (ERP) restrictions which exist for the unlicenced service. This technology will be the ticket to developing high-speed wireless LANs and MANs which, using the Internet as a backbone, could finally realize the dream of a high-performance wide-area AMPRnet which can support

the applications (WWW, audio and video conferencing, etc.) that get people excited about computer networking these days.

Although the dream as stated above is somewhat controversial, the author believes it represents the best hope of attracting new people to the hobby, providing a basis for experimentation and training in state-of-the-art wireless techniques and networking, and, ultimately, retaining spectrum for the amateur radio service. One problem is that most of the people attracted to using digital wireless techniques have little or no background in RF. When it comes to setting up wireless links which will work over some distance, they tend to lack the necessary knowledge about antennas, transmission lines and, especially, the subtleties of radio propagation. This paper deals with the latter area, in the hopes of providing this new crop of digital experimenters with some tools to help them build wireless links which work.

The main emphasis of this paper is on predicting the path loss of a link, so that one can approach the installation of the antennas and other RF equipment with some degree of confidence that the link will work. The focus is on acquiring a feel for radio propagation, and pointing the way towards recognizing the alternatives that may exist and the instances in which experimentation may be fruitful. We'll also look at some propagation aspects which are of particular relevance to digital signaling.

Estimating Path Loss

The fundamental aim of a radio link is to deliver sufficient signal power to the receiver at the far end of the link to achieve some performance objective. For a data transmission system, this objective is usually specified as a minimum bit error rate (BER). In the receiver demodulator, the BER is a function of the signal to noise ratio (SNR). At the frequencies under consideration here, the noise power is often dominated by the internal receiver noise; however, this is not always the case, especially at the lower (VHF) end of the range. In addition, the "noise" may also include significant power from interfering signals, necessitating the delivery of higher signal power to the receiver than would be the case under more ideal circumstances (i.e., back-to-back through an attenuator). If the channel contains multipath, this may also have a major impact on the BER. We will consider multipath in more detail later - for now, we will focus on predicting the signal power which will be available to the receiver.

Free Space Propagation

The benchmark by which we measure the loss in a transmission link is the loss that would be expected in free space - in other words, the loss that would occur in a region which is free of all objects that might absorb or reflect radio energy. This represents the ideal case which we hope to approach in our real world radio link (in fact, it is possible to have path loss which is less than the "free space" case, as we shall see later, but it is far more common to fall short of this goal).

Calculating free space transmission loss is quite simple. Consider a transmitter with power P_t coupled to an antenna which radiates equally in all directions (everyone's favorite mythical antenna, the *isotropic* antenna). At a distance d from the transmitter, the radiated power is distributed uniformly over an area of $4\pi d^2$ (i.e. the surface area of a sphere of radius d), so that the *power flux density* is:

$$s = \frac{P_t}{4\pi d^2} \quad (1)$$

The transmission loss then depends on how much of this power is captured by the receiving antenna. If the capture area, or *effective aperture* of this antenna is A,, then the power which can be delivered to the receiver (assuming no mismatch or feedline losses) is simply

$$P_r = sA_r \quad (2)$$

For the hypothetical isotropic receiving antenna, we have

$$A_r = \frac{\lambda^2}{4\pi} \quad (3)$$

Combining equations (1) and (3) into (2), we have

$$P_r = P_t\left(\frac{\lambda}{4\pi d}\right)^2 \quad (4)$$

The free space path loss between isotropic antennas is P_t / P_r. Since we usually are dealing with frequency rather than wavelength, we can make the substitution $\lambda = c/f$ to get

$$L_p = \left(\frac{4\pi}{c}\right)^2 f^2 d^2 \quad (5)$$

This shows the classic square-law dependence of signal level versus distance. What troubles some people when they see this equation is that the path loss also increases as the square of the frequency. Does this mean that the transmission medium is inherently more lossy at higher frequencies? While it is true that absorption of RF by various materials (buildings, trees, water vapor, etc.) tends to increase with frequency, remember we are talking about "free space" here. The frequency dependence in this case is solely due to the decreasing effective aperture of the receiving antenna as the frequency increases. This is intuitively reasonable, since the physical size of a given antenna type is inversely proportional to frequency. If we double the frequency, the linear dimensions of the antenna decrease by a factor of one-half, and the capture area by a factor of one-quarter. The antenna therefore captures only one-quarter of the power flux density at the higher frequency versus the lower one, and delivers 6 dB less signal to the receiver. However, in most cases we can easily get this 6 dB back by increasing the effective aperture, and hence the gain, of the receiving antenna. For example, suppose we are using a parabolic dish antenna at the lower frequency. When we double the frequency, instead of allowing the dish to be scaled down in size so as to produce the same gain as before, we can maintain the same reflector size. This gives us the same effective aperture as before (assuming that the feed is properly redesigned for the new frequency, etc.), and 6 dB more gain (remembering that the gain is with respect to an isotropic or dipole reference antenna at the *same frequency*). Thus the free space path loss is now the same at both frequencies; moreover, if we maintained the same physical aperture at *both* ends of the link, we would actually have 6 dB *less* path loss at the higher frequency. You can picture this in terms of being able to

focus the energy more tightly at the frequency with the shorter wavelength. It has the added benefit of providing greater discrimination against multipath - more about this later.

The free space path loss equation is more usefully expressed logarithmically:

$$L_p = 32.4 + 20 \log f + 20 \log d \text{ dB} \quad \text{(f in MHz, d in km)} \tag{6a}$$

or

$$L_p = 36.6 + 20 \log f + 20 \log d \text{ dB} \quad \text{(f in MHz, d in miles)} \tag{6b}$$

This shows more clearly the relationship between path loss and distance: path loss increases by 20 dB/decade or 6 dB/octave, so each time you double the distance, you lose another 6 dB of signal under free space conditions.

Of course, in looking at a real system, we must consider the actual antenna gains and cable losses in calculating the signal power P_r which is available at the receiver input:

$$P_r = P_t - L_p + G_t + G_r - L_t - L_r \tag{7}$$

where P_t = transmitter power output (dBm or dBW, same units as P_r)
L_p = free space path loss between isotropic antennas (dB)
G_t = transmit antenna gain (dBi)
G_r = receive antenna gain (dBi)
L_t = transmission line loss between transmitter and transmit antenna (dB)
L_r = transmission line loss between receive antenna and receiver input (dB)

A table of transmission line losses for various bands and popular cable types can be found in the Appendix.

Example 1. Suppose you have a pair of 915 MHz WaveLAN cards, and want to use them on a 10 km link on which you believe free space path loss conditions will apply. The transmitter power is 0.25 W, or +26 dBm. You also have a pair of yagi antennas with 10 dBi gain, and at each end of the link, you need about 50 ft (15 m) of transmission line to the antenna. Let's say you're using LMR-400 coaxial cable, which will give you about 2 dB loss at 915 MHz for each run. Finally, the path loss from equation (6a) is calculated, and this gives 111.6 dB, which we'll round off to 112 dB. The expected signal power at the receiver is then, from (7):

$$P_r = 26 - 112 + 10 + 10 - 2 - 2 = -70 \text{ dBm}$$

According to the WaveLAN specifications, the receivers require -78 dBm signal level in order to deliver a low bit error rate (BER). So, we should be in good shape, as we have 8 dB of margin over the minimum requirement. However, this will only be true *if* the path really is equivalent to the free space case, and this is a big *if*! We'll look at means of predicting whether the free space assumption holds in the next section.

Path Loss on Line of Sight Links

The term Line of Sight (LOS) as applied to radio links has a pretty obvious meaning: the antennas at the ends of the link can "see" each other, at least in a radio sense. In many cases, radio LOS equates to optical LOS: you're at the location of the antenna at one end of the link, and with the unaided eye or binoculars, you can see the antenna (or its future site) at the other end of the link. In other cases, we may still have an LOS path even though we can't see the other end visually. This is because the radio horizon extends beyond the optical horizon. Radio waves follow slightly curved paths in the atmosphere, but if there is a direct path between the antennas which doesn't pass through any obstacles, then we still have radio LOS. Does having LOS mean that the path loss will be equal to the free space case which we have just considered? In some cases, the answer is yes, but it is definitely not a sure thing. There are three mechanisms which may cause the path loss to differ from the free space case:

- *refraction* in the earth's atmosphere, which alters the trajectory of radio waves, and which can change with time.
- *diffraction* effects resulting from objects near the direct path.
- *reflections* from objects, which may be either near or far from the direct path.

We examine these mechanisms in the next three sections.

Atmospheric Refraction

As mentioned previously, radio waves near the earth's surface do not usually propagate in precisely straight lines, but follow slightly curved paths. The reason is well-known to VHF/UHF DXers: refraction in the earth's atmosphere. Under normal circumstances, the index of refraction decreases monotonically with increasing height, which causes the radio waves emanating from the transmitter to bend slightly downwards towards the earth's surface instead of following a straight line. The effect is more pronounced at radio frequencies than at the wavelength of visible light, and the result is that the radio waves can propagate beyond the optical horizon, with no additional loss other than the free space distance loss. There is a convenient artifice which is used to account for this phenomenon: when the path profile is plotted, we reduce the curvature of the earth's surface. If we choose the curvature properly, the paths of the radio waves can be plotted as straight lines. Under normal conditions, the gradient in refractivity index is such that real world propagation is equivalent to straight-line propagation over an earth whose radius is greater than the real one by a factor of 4/3 - thus the often-heard term "4/3 earth radius" in discussions of terrestrial propagation. However, this is just an approximation that applies under typical conditions - as VHF/UHF experimenters well know, unusual weather conditions can change the refractivity profile dramatically. This can lead to several different conditions. In *superrefraction*, the rays bend more than normal and the radio horizon is extended; in extreme cases, it leads to the phenomenon known as *ducting*, where the signal can propagate over enormous distances beyond the normal horizon. This is exciting for DXers, but of little practical use for people who want to run data links. The main consequence for digital experimenters is that they may occasionally experience interference from unexpected sources. A more serious concern is *subrefraction*, in which the bending of the rays is less than normal, thus shortening the radio horizon and reducing the clearance over obstacles along the path. This may lead to increased path loss, and possibly even an outage. In commercial radio

link planning, the statistical probability of these events is calculated and allowed for in the link design (distance, path clearance, fading margin, etc.). We won't get into all of the details here; suffice it to say that reliability of your link will tend to be higher if you back off the distance from the maximum which is dictated by the normal radio horizon. Not that you shouldn't try and stretch the limits when the need arises, but a link which has optical clearance is preferable to one which doesn't. It's also a good idea to build in some margin to allow for fading due to unusual propagation situations, and to allow as much clearance from obstacles along the path as possible. For short-range links, the effects of refraction can usually be ignored.

Diffraction and Fresnel Zones

Refraction and reflection of radio waves are mechanisms which are fairly easy to picture, but diffraction is much less intuitive. To understand diffraction, and radio propagation in general, it is very helpful to have some feeling for how radio waves behave in an environment which is not strictly "free space". Consider Fig. 1, in which a wavefront is traveling from left to right, and encountering an obstacle which absorbs or reflects all of the incident radio energy. Assume that the incident wavefront is uniform; i.e., if we measure the field strength along the line A-A', it is the same at all points. Now, what will be the field strength along a line B-B' on the other side of the obstacle? To quantify this, we provide an axis in which zero coincides with the top of the obstacle, and negative and positive numbers denote positions above and below this, respectively (we'll define the parameter v used on this axis a bit later).

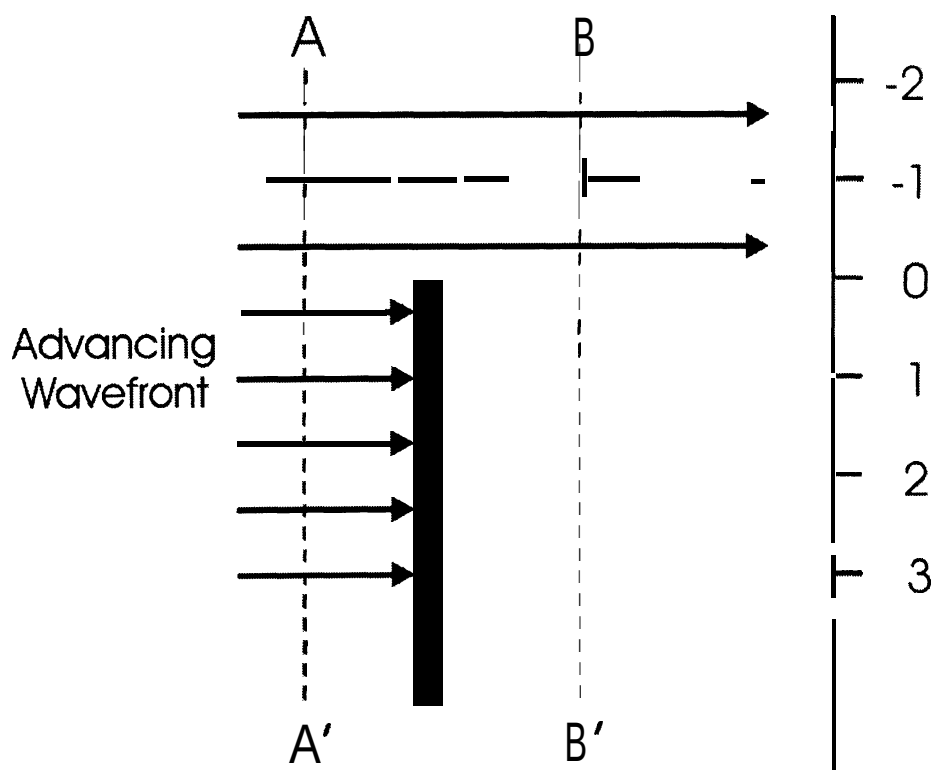


Figure 1 Shadowing of Radio Waves by an Object

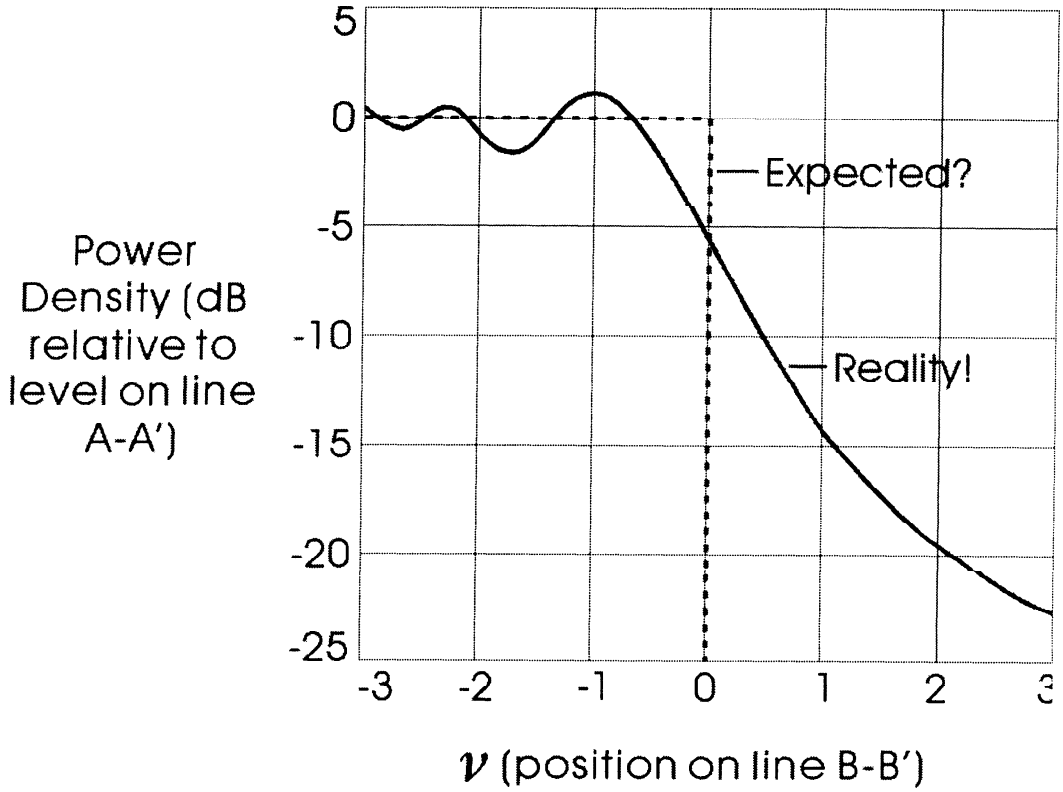


Figure 2 Signal Levels on the Far Side of the Shadowing Object

Intuition may lead one to expect the field strength along B-B' to look like the dashed line in Fig. 2, with complete shadowing and zero signal below the top of the obstacle, and no effect at all above it. The solid line shows the reality: not only does energy "leak" into the shadowed area, but the field strength above the top of the obstacle is also disturbed. At a position which is level with the top of the obstacle, the signal power density is down by some 6 dB, despite the fact that this point is in "line of sight" of the source. This effect is less surprising when one considers other familiar instances of wave motion. Picture, for example, tossing a rock in a pond and watching the ripples propagate outward. When they encounter an object such as a boat or a pier, you will see that the water behind the object is also disturbed, and that the waves traveling past, but close to, the object are also affected somewhat. Similarly, consider a distant source of sound waves: if the sound level is well above the ambient level, then moving behind an object which absorbs the incident sound energy completely does not result in the sound disappearing completely - it is still audible at a lower level, due to diffraction (as an aside, it is interesting to note that the wavelength of a 1 KHz sound wave is nearly the same as a 1 GHz radio wave). So much for analogies - let's get back to radio waves.

The explanation for the non-intuitive behavior of radio waves in the presence of obstacles which appear in their path is found in something called *Huygens ' Principle.* Huygens showed that propagation occurs as follows: each point on a wavefront acts as a source of a secondary wavefront known as a *wavelet*, and a new wavefront is then built up from the combination of the contributions frorn all of the wavelets on the preceding wavefront. The secondary wavelets do not radiate equally in all directions - their

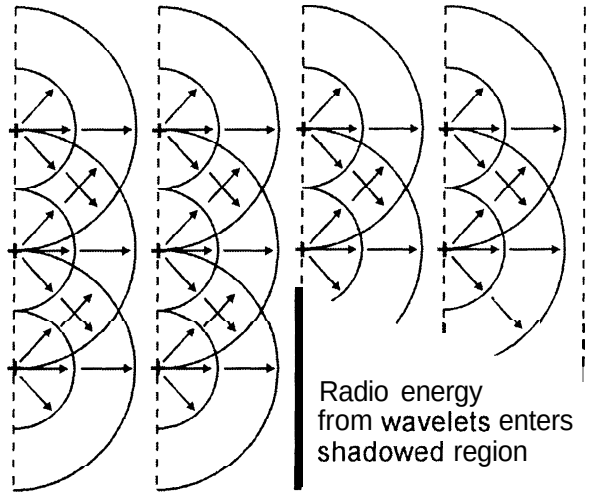


Figure 3 Representation of Radio Waves as Wavelets

amplitude in a given direction is proportional to (I + cos a), where a is the angle between that direction and the direction of propagation of the wavefront. The amplitude is therefore maximum in the direction of propagation (i.e., normal to the wavefront), and zero in the reverse direction. The representation of a wavefront as a collection of wavelets is shown in Fig. 3.

At a given point on the new wavefront (point B), the signal vector (phasor) is determined by vector addition of the contributions from the wavelets on the preceding wavefront, as shown in Fig. 4. The largest component is from the nearest wavelet, and we then get symmetrical contributions from the points above and below it. These latter vectors are shorter, due to the angular reduction of amplitude

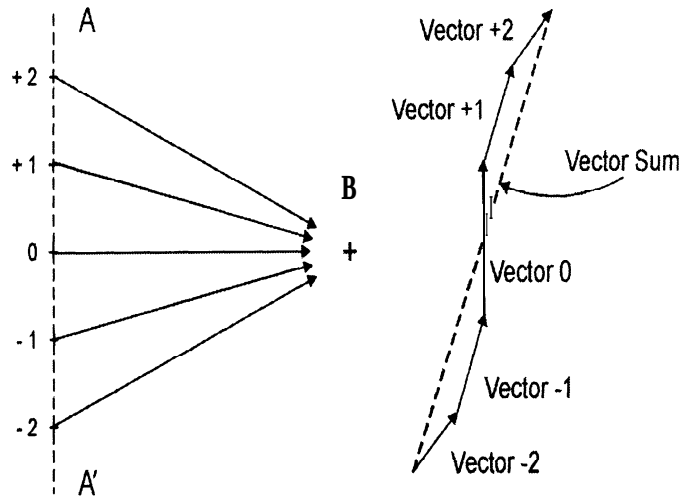


Figure 4 Building of a New Wavefront by Vector Summation

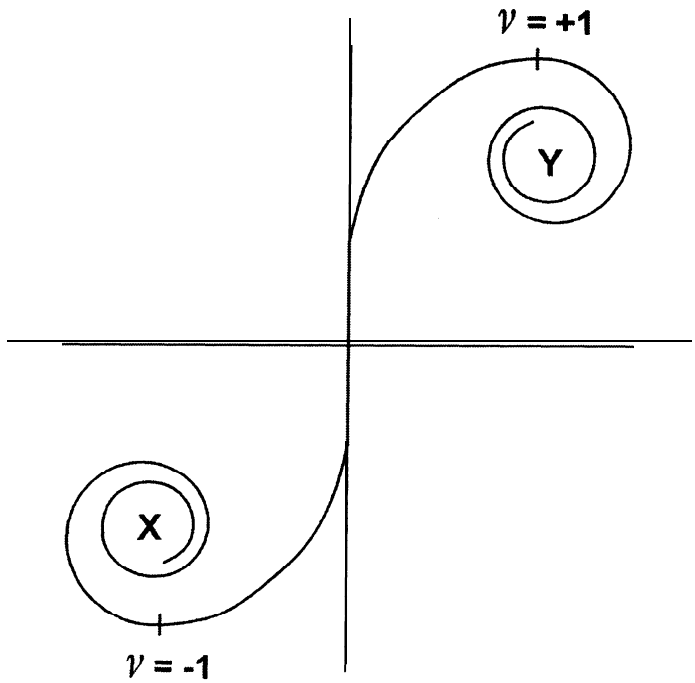


Figure 5 The Cornu Spiral

mentioned above, and also the greater distance traveled. The greater distance also introduces more time delay, and hence the rotation of the vectors as shown in the figure. As we include contributions from points farther and farther away, the corresponding vectors continue to rotate and diminish in length, and they trace out a double-sided spiral path, known as the Cornu spiral.

The Cornu spiral, shown in Fig. 5, provides the tool we need to visualize what happens when radio waves encounter an obstacle. In free space, at every point on a new wavefront, all contributions from the wavelets on the preceding wavefront are present and unattenuated, so the resultant vector corresponds to the complete spiral (i.e., the endpoints of the vector are X and Y). Now, consider again the situation shown in Fig. 1, and for each location on the wavefront B-B', visualize the makeup of the Cornu spiral (note that the top of the obstacle is assumed to be sufficiently narrow that no significant reflections can occur from it). At position 0, level with the top of the obstacle, we will have only contributions from the positive half of the preceding wavefront at A-A', since all of the others are blocked by the obstacle. Therefore, the received components form only the upper half of the spiral, and the resultant vector is exactly half the length of the free space case, corresponding to a 6 dB reduction in amplitude. As we go lower on the line B-B', we start to get blockage of components from the positive side of the A-A' wavefront, removing more and more of the vectors as we go, and leaving only the tight upper spiral. The resulting amplitude diminishes monotonically towards zero as we move down the new wavefront, but there *is* still signal present at all points behind the obstacle, as shown in the graph in Fig. 2. How about the points along line B-B' *above* the obstacle, where the graph shows those mysterious ripples? Again look at the Cornu spiral: as we move up the line, we begin to add contributions from the negative side of the A-A' wavefront (vectors -1, -2, etc.). Note what happens to the resultant vector - as we make the first turn around the bottom of the spiral, it reaches its maximum length, corresponding to

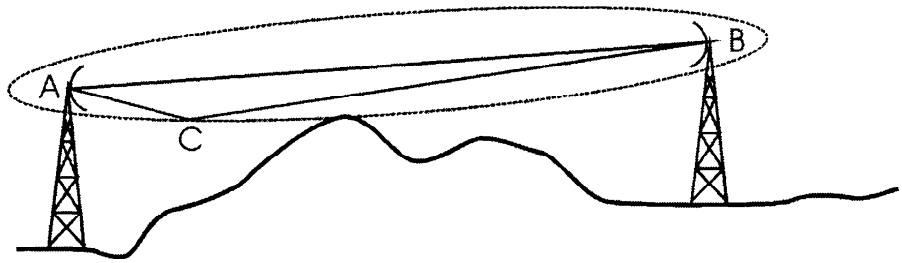


Figure 6 Fresnel Zone for a Radio Link

the highest peak in the graph of Fig. 2. As we continue to move up B-B' and add more components, we swing around the spiral and reach the minimum length for the resultant vector (minimum distance from point Y). Further progression up B-B' results in further motion around the spiral, and the amplitude of the resultant oscillates back and forth, with the amplitude of the oscillation steadily decreasing as the resultant converges on the free space value, given by the complete Cornu spiral (vector X-Y).

So, in a nutshell, to visualize what happens to radio waves when they encounter an obstacle, we have to develop a picture of the wavefront after the obstacle as a function of the wavefront just before it (as opposed to simply tracing rays from the distant source). Now we're in a position to talk about Fresnel zones. A Fresnel zone is a simpler concept once you have some understanding of diffraction: it is the volume of space enclosed by an ellipsoid which has the two antennas at the ends of a radio link at its foci. The two-dimensional representation of a Fresnel zone is shown in Fig. 6. The surface of the ellipsoid is defined by the path ACB, which exceeds the length of the direct path AB by some fixed amount. This amount is $n\lambda/2$, where n is a positive integer. For the first Fresnel zone, $n = 1$ and the path length differs by h/2 (i.e., a 180° phase reversal with respect to the direct path). For most practical purposes, only the first Fresnel zone need be considered. A radio path has *first Fresnel zone clearance* if, as shown in Fig. 6, no objects capable of causing significant diffraction penetrate the corresponding ellipsoid. What does this mean in terms of path loss? Recall how we constructed the wavefront behind an object by vector addition of the wavelets comprising the wavefront in front of the object, and apply this to the case where we have exactly first Fresnel zone clearance. We wish to find the strength of the direct path signal after it passes the object. Assuming there is only one such object near the Fresnel zone, we can look at the resultant wavefront at the destination point B. In terms of the Cornu spiral, the upper half of the spiral is intact, but part of the lower half is absent, due to blockage by the object. Since we have exactly first Fresnel clearance, the final vector which we add to the bottom of the spiral is 180" out of phase with the direct-path vector - i.e., it is pointing downwards. This means that we have passed the bottom of the spiral and are on the way back up, and the resultant vector is near the free space magnitude (a line between X and Y in Fig. 5). In fact, it is sufficient to have 60% of the first Fresnel clearance, since this will still give a resultant which is very close to the free space value.

In order to quantify diffraction losses, they are usually expressed in terms of a dimensionless parameter v, given by:

$$v = 2\sqrt{\frac{Ad}{\lambda}} \qquad (8)$$

where Ad is the difference in lengths of the straight-line path between the endpoints of the link and the path which just touches the tip of the diffracting object (see Fig. 7, where Ad = $d_1 + d_2 - d$). By convention, v is positive when the direct path is blocked (i.e., the obstacle has positive height), and negative when the direct path has some clearance ("negative height"). When the direct path just grazes the object, v = 0. This is the parameter shown in Figures 1 and 2. Since in this section we are considering LOS paths, this corresponds to specifying that $v \leq 0$. For first Fresnel zone clearance, we have Ad = $\lambda/2$, so from equation (8), v = -1.4. From Fig. 2, we can see that this is more clearance than necessary - in fact, we get slightly higher signal level (and path loss less than the free space value) if we reduce the clearance to v = -1, which corresponds to Ad = h/4. The v = -1 point is also shown on the Cornu spiral in Fig. 5. Since Ad= $\lambda/4$, the last vector added to the summation is rotated 90° from the direct-path vector, which brings us to the lowest point on the spiral. The resultant vector then runs from this point to the upper end of the spiral at point Y. It's easy to see that this vector is a bit longer than the distance from X to Y, so we have a slight gain (about 1.2 dB) over the free space case. We can also see how we can back off to 60% of first Fresnel zone clearance ($v \approx -0.85$) without suffering significant loss.

But how do we calculate whether we have the required clearance? The geometry for Fresnel zone calculations is shown in Fig. 7. Keep in mind that this is only a two-dimensional representation, but Fresnel zones are three-dimensional. The same considerations apply when the objects limiting path clearance are to the side or even above the radio path. Since we are considering LOS paths in this section, we are dealing only with the "negative height" case, shown in the lower part of the figure. We will look at the case where h is positive later, when we consider non-LOS paths.

For first Fresnel zone clearance, the distance h from the nearest point of the obstacle to the direct path must be at least

$$h = 2\sqrt{\frac{\lambda\, d_1 d_2}{d_1 + d_2}} \qquad (9)$$

where d_1 and d_2 are the distances from the tip of the obstacle to the two ends of the radio circuit. This formula is an approximation which is not valid very close to the endpoints of the circuit. For convenience, the clearance can be expressed in terms of frequency:

$$h = 17.3\sqrt{\frac{d_1 d_2}{f(d_1 + d_2)}} \qquad (10a)$$

where f is the frequency in GHz, d_1 and d_2 are in km, and h is in meters. Or:

$$h = 72.1\sqrt{\frac{d_1 d_2}{f(d_1 + d_2)}} \qquad (10b)$$

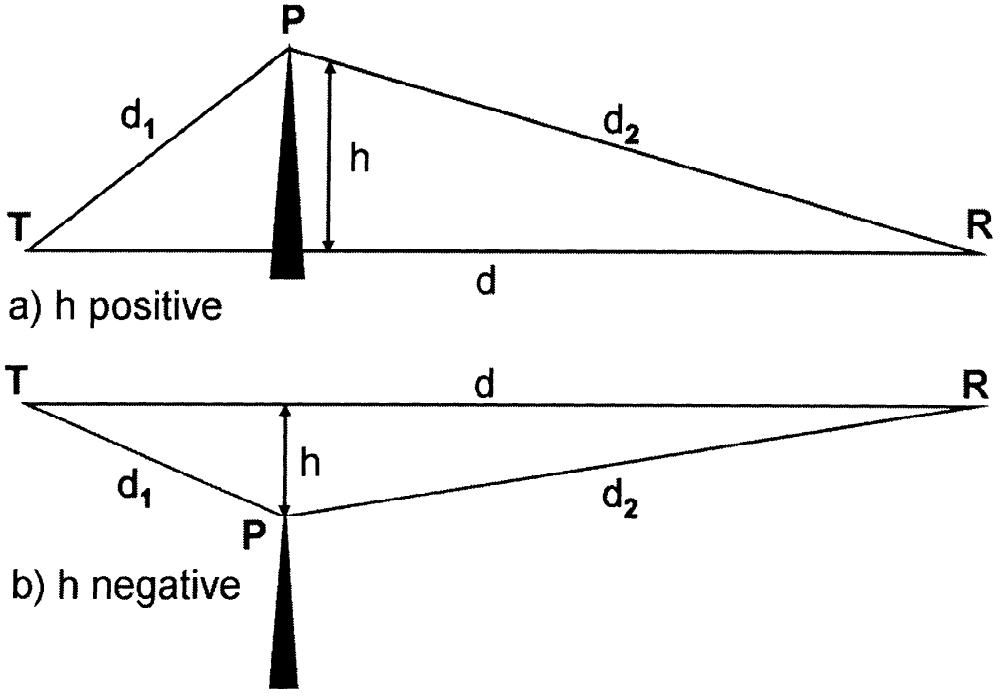


Figure 7 Fresnel Zone Geometry

where f is in GHz, d_1 and d_2 in statute miles, and h is in feet.

Example 2. We have a 10 km LOS path over which we wish to establish a link in the 915 MHz band. The path profile indicates that the high point on the path is 3 km from one end, and the direct path clears it by about 18 meters (60 ft.) - do we have adequate Fresnel zone clearance? From equation (10a), with d_1 = 3 km, d_2 = 7 km, and f = 0.915 GHz, we have h = 26.2 m for first Fresnel zone clearance (strictly speaking, h = -26.2 m). A clearance of 18 m is about 70% of this, so it is sufficient to allow negligible diffraction loss).

Fresnel zone clearance may not seem all that important - after all, we said previously that for the zero clearance (grazing) case, we have 6 dB of additional path loss. If necessary, this could be overcome with, for example, an additional 3 dB of antenna gain at each end of the circuit. Now it's time to confess that the situation depicted in Figures 1 and 2 is a special case, known as "knife edge" diffraction. Basically, this means that the top of the obstacle is small in terms of wavelengths. This is sometimes a reasonable approximation of an object in the real world, but more often than not, the obstacle will be rounded (such as a hilltop) or have a large flat surface (like the top of a building), or otherwise depart from the knife edge assumption. In such cases, the path loss for the grazing case can be considerably more than 6 dB - in fact, 20 dB would be a better estimate in many cases. So, Fresnel zone clearance can be pretty important on real-world paths. And, again, keep in mind that the Fresnel zone is three-dimensional, so clearance must also be maintained from the sides of buildings, etc. if path loss is to be minimized. Another point to consider is the effect on Fresnel zone clearance of changes in atmospheric refraction, as discussed in the last section. We may have adequate clearance on a longer path under

normal conditions (i.e., 4/3 earth radius), but lose the clearance when unusual refraction conditions prevail. On longer paths, therefore, it is common in commercial radio links to do the Fresnel zone analysis on something close to "worst case" rather than typical refraction conditions, but this may be less of a concern in amateur applications.

Most of the material in this section was based on Ref. [2], which is highly recommended for further reading.

Ground Reflections

An LOS path may have adequate Fresnel zone clearance, and yet still have a path loss which differs significantly from free space under normal refraction conditions. If this is the case, the cause is probably multipath propagation resulting from reflections (multipath also poses particular problems for digital transmission systems - we'll look at this a bit later, but here we are only considering path loss).

One common source of reflections is the ground. It tends to be more of a factor on paths in rural areas; in urban settings, the ground reflection path will often be blocked by the clutter of buildings, trees, etc. In paths over relatively smooth ground or bodies of water, however, ground reflections can be a major determinant of path loss. For any radio link, it is worthwhile to look at the path profile and see if the ground reflection has the potential to be significant. It should also be kept in mind that the reflection point is not at the midpoint of the path unless the antennas are at the same height and the ground is not sloped in the reflection region -just the remember the old maxim from optics that the angle of incidence equals the angle of reflection.

Ground reflections can be good news or bad news, but are more often the latter. In a radio path consisting of a direct path plus a ground-reflected path, the path loss depends on the relative amplitude and phase relationship of the signals propagated by the two paths. In extreme cases, where the ground-reflected path has Fresnel clearance and suffers little loss from the reflection itself (or attenuation from trees, etc.), then its amplitude may approach that the direct path. Then, depending on the relative phase shift of the two paths, we may have an enhancement of up to 6 dB over the direct path alone, or cancellation resulting in additional path loss of 20 dB or more. If you are acquainted with Mr. Murphy, you know which to expect! The difference in path lengths can be estimated from the path profile, and then translated into wavelengths to give the phase relationship. Then we have to account for the reflection itself, and this is where things get interesting. The amplitude and phase of the reflected wave depend on a number of variables, including conductivity and permittivity of the reflecting surface, frequency, angle of incidence, and polarization

It is difficult to summarize the effects of all of the variables which affect ground reflections, but a typical case is shown in Fig. 8 [2]. This particular figure is for typical ground conditions at 100 MHz, but the same behavior is seen over a wide range of ground constants and frequencies. Notice that there is a large difference in reflection amplitudes between horizontal and vertical polarization (denoted on the curves with "h" and "v", respectively), and that vertical polarization in general gives rise to a much smaller reflected wave. However, the difference is large only for angles of incidence greater than a few degrees (note that, unlike in optics, in radio transmission the angle of incidence is normally measured with respect to a tangent to the reflecting surface rather than a normal to it); in practice, these angles will only occur on very short paths, or paths with extraordinarily high antennas. For typical paths, the angle of

incidence tends to be of the order of one degree or less - for example, for a 10 km path over smooth earth with 10 m antenna heights, the angle of incidence of the ground reflection would only be about 0.11 degrees. In such a case, both polarizations will give reflection amplitudes near unity (i.e., no reflection loss). Perhaps more surprisingly, there will also be a phase reversal in both cases. Horizontally-polarized waves always undergo a phase reversal upon reflection, but for vertically-polarized waves, the phase change is a function of the angle of incidence and the ground characteristics.

The upshot of all this is that for most paths in which the ground reflection is significant (and no other reflections are present), there will be very little difference in performance between horizontal and vertical polarization. For very short paths, horizontal polarization will generally give rise to a stronger reflection. If it turns out that this causes cancellation rather than enhancement, switching to vertical polarization may provide a solution. In other words, for shorter paths, it is usually worthwhile to try both polarizations to see which works better (of course, other factors such as mounting constraints and rejection of other sources of multipath and interference also enter into the choice of polarization).

As stated above, for either polarization, as the path gets longer we approach. the case where the ground reflection produces a phase reversal and very little attenuation. At the same time, the direct and reflected paths are becoming more nearly equal. The path loss ripples up and down as we increase the distance, until we reach the point where the path lengths differ by just one-half wavelength. Combined with the 180° phase shift caused by the ground reflection, this brings the direct and reflected signals into phase, resulting in an enhancement over the free space path loss (theoretically 6 dB, but this will seldom be

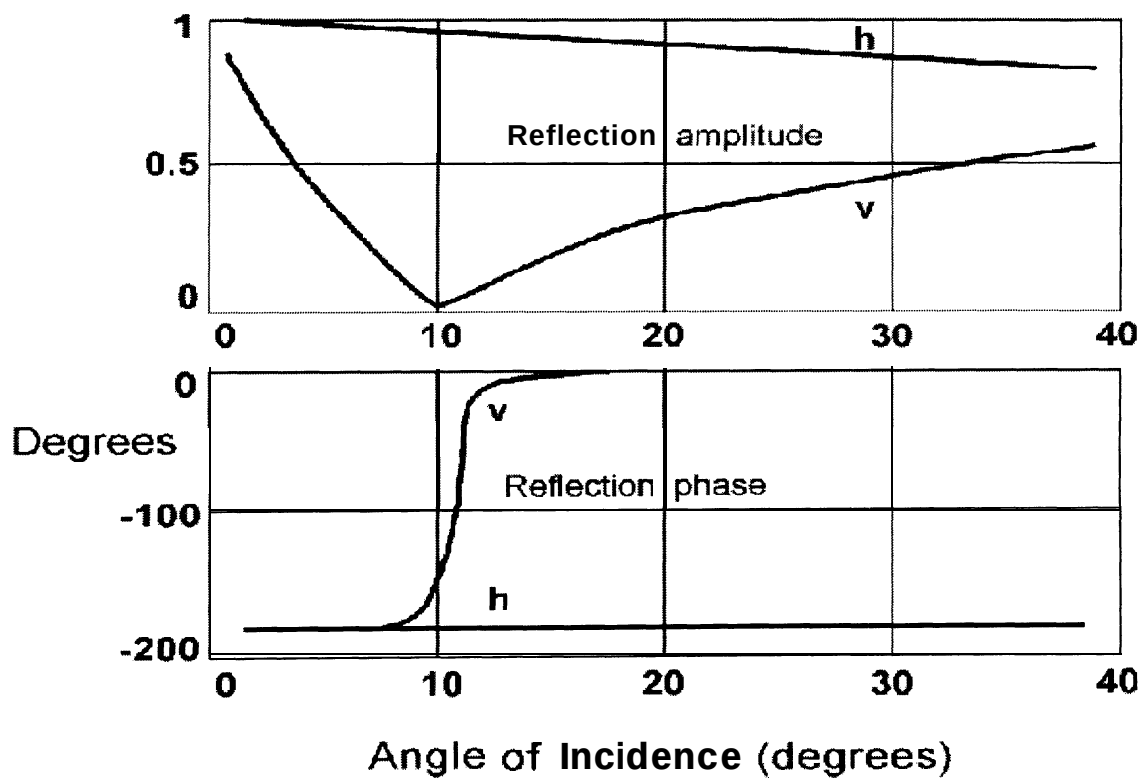


Figure 8 Typical Ground Reflection Parameters

realized in practice). Thereafter, it's all downhill as the distance is further increased, since phase difference between the two paths approaches in the limit the 180° phase shift of the ground reflection. It can be shown that, in this region, the received power follows an inverse fourth-power law as a function of distance instead of the usual square law (i.e., 12 dB more attenuation when you double the distance, instead of 6 dB). The distance at which the path loss starts to increase at the fourth-power rate is reached when the ellipsoid corresponding to the first Fresnel zone just touches the ground. A reasonably good estimate of this distance can be calculated from the equation

$$d = \frac{4h_1h_2}{\lambda} \qquad (11)$$

where h_1 and h_2 are the antenna heights above the ground reflection point. For example, for antenna heights of 10 m, at 915 MHz (λ = 33 cm) we will be into the fourth-law loss region for links longer than about 1.2 km.

So, for longer-range paths, ground reflections are always bad news. Serious problems with ground reflections are most commonly encountered with radio links across bodies of water. Spread spectrum techniques and diversity antenna arrangements usually can't overcome the problems - the solution lies in siting the antennas (e.g., away from the shore of the body of water) such that the reflected path is cut off by natural obstacles, while the direct path is unimpaired. In other cases, it may be possible to adjust the antenna locations so as to move the reflection point to a rough area of land which scatters the signal rather than creating a strong specular reflection.

Other Sources of Reflections

Much of what has been said about ground reflections applies to reflections from other objects as well. The "ground reflection" on a particular path may be from a building rooftop rather than the ground itself, but the effect is much the same. On long links, reflections from objects near the line of the direct path will almost always cause increased path loss - in essence, you have a permanent "flat fade" over a very wide bandwidth. Reflections from objects which are well off to the side of the direct path are a different story, however. This is a frequent occurrence in urban areas, where the sides of buildings can cause strong reflections. In such cases, the angle of incidence may be much larger than zero, unlike the ground reflection case. This means that horizontal and vertical polarization may behave quite differently - as we saw in Fig. 8, vertically polarized signals tend to produce lower-amplitude reflections than horizontally polarized signals when the angle of incidence exceeds a few degrees. When the reflecting surface is vertical, like the side of a building, a signal which is transmitted with horizontal polarization effectively has vertical polarization as far as the reflection is concerned. Therefore, horizontal polarization will generally result in weaker reflections and less multipath than vertical polarization in these cases.

Effects of Rain, Snow and Fog

The loss of LOS paths may sometimes be affected by weather conditions (other than the refraction effects which have already been mentioned). Rain and fog (clouds) become a significant source of attenuation only when we get well into the microwave region. Attenuation from fog only becomes noticeable (i.e., attenuation of the order of 1 dB or more) above about 30 GHz. Snow is in this category as well. Rain attenuation becomes significant at around 10 GHz, where a heavy rainfall may cause additional path loss of the order of 1 dB/km.

Path Loss on Non-Line of Sight Paths

We have spent quite a bit of time looking at LOS paths, and described the mechanisms which often cause them to have path loss which differs from the "free space" assumption. We've seen that the path loss isn't always easy to predict. When we have a path which is not LOS, it becomes even more difficult to predict how well signals will propagate over it. Unfortunately, non-LOS situations are sometimes unavoidable, particularly in urban areas. The following sections deal with some of the major factors which must be considered.

Diffraction Losses

In some special cases, such as diffraction over a single obstacle which can be modeled as a knife edge, the loss of a non-LOS path can be predicted fairly readily. In fact, this is the same situation that we saw in Figures 1 and 2, with the diffraction parameter v > 0. This parameter, from equation (8), is

$$v = 2\sqrt{\frac{\Delta d}{\lambda}}$$

To get Ad, measure the straight-line distance between the endpoints of the link. Then measure the length of the actual path, which includes the two endpoints and the tip of the knife edge, and take the difference between the two. The geometry is shown in Fig. 7(a), the "positive h" case. A good approximation to the knife-edge diffraction loss in dB can then be calculated from

$$L(v) = 6.9 + 20 \log\left[\sqrt{v^2 + 1} + v\right] \qquad (12)$$

Example 3. We want to run a 915 MHz link between two points which are a straight-line distance of 25 km apart. However, 5 km from one end of the link, there is a ridge which is 100 meters higher than the two endpoints. Assuming that the ridge can be modeled as a knife edge, and that the paths from the endpoints to the top of ridge are LOS with adequate Fresnel zone clearance, what is the expected path loss? From simple geometry, we find that length of the path over the ridge is 25,001.25 meters, so that Ad = 1.25 m. Since λ = 0.33 m, the parameter v, from (8), is 3.89. Substituting this into (12), we find that the expected diffraction loss is 24.9 dB. The free space path loss for a 25 km path at 915 MHz is, from equation (6a), 119.6 dB, so the total predicted path loss for this path is 144.5 dB. This is too lossy a path for many WLAN devices. For example, suppose we are using WaveLAN cards with 13 dBi gain antennas, which (disregarding feedline losses) brings them up to the maximum allowable EIRP of +36 dBm. This will produce, at the antenna terminals at the other end of the link, a received power of (36 - 144.5 + 13) = -95.5 dBm. This falls well short of the -78 dBm requirement of the WaveLAN cards. On the other hand, a lower-speed system may be quite usable over this path. For instance, the FreeWave 115 Kbps modems require only about - 108 dBm for reliable operation, which is a comfortable margin below our predicted signal levels.

To see the effect of operating frequency on diffraction losses, we can repeat the calculation, this time using 144 MHz, and find the predicted diffraction loss to be 17.5 dB, or 7.4 dB less than at 915 MHz. At 2.4 GHz, the predicted loss is 29.0 dB, an increase of 4.1 dB over the 915 MHz case (these differences are for the diffraction losses only, not the only total path loss).

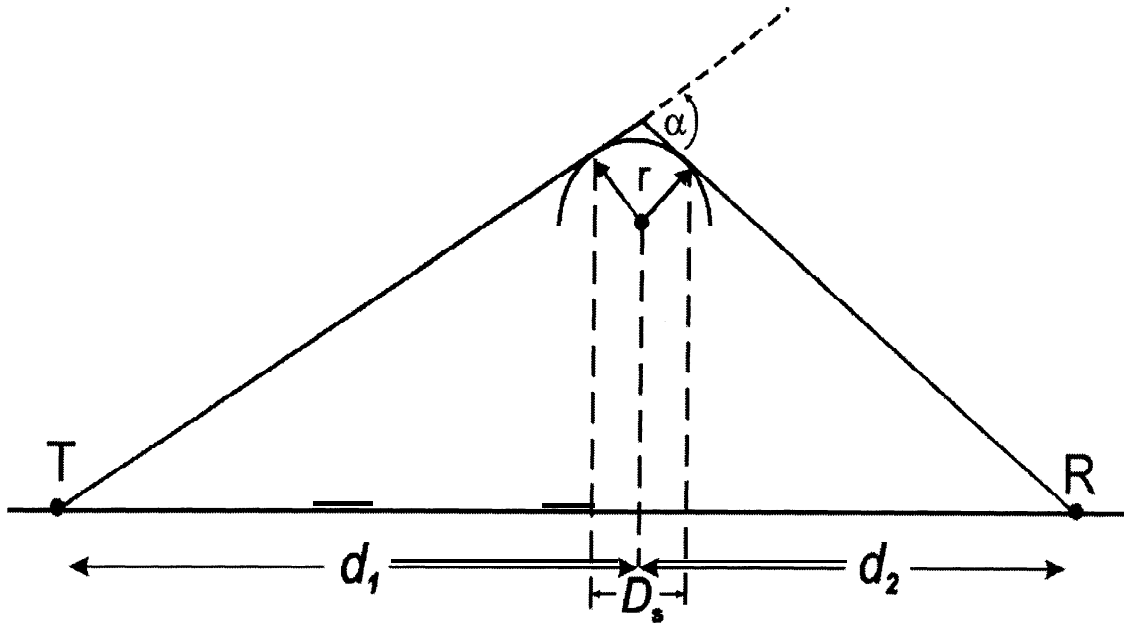


Figure 9 Diffraction by a Rounded Obstacle

Unfortunately, the paths which digital experimenters are faced with are seldom this simple. They will frequently involve diffraction over multiple rooftops or other obstacles, many of which don't resemble knife edges. The path losses will generally be substantially greater in these cases than predicted by the single knife edge model. The paths will also often pass through objects such as trees and wood-frame buildings which are semi-transparent at radio frequencies. Many models have been developed to try and predict path losses in these more complex cases. The most successful are those which deal with restricted scenarios rather than trying to cover all of the possibilities. One common scenario is diffraction over a single obstacle which is too rounded to be considered a knife edge. There are different ways of treating this problem; the one described here is from Ref. [3]. The top of the object is modeled as a cylinder of radius r, as shown in Fig. 9. To calculate the loss, you need to plot the profile of the actual object, and then draw straight lines from the link endpoints such that they just graze the highest part of the object as seen from their individual perspectives. Then the parameters D_s, d_1, d_2 and a are estimated, and an estimate of the radius r can then be calculated from

$$r = \frac{2D_s d_1 d_2}{\alpha\,(d_1^2 + d_2^2)} \tag{13}$$

Note that the angle a is measured in radians. The procedure then is to calculate the knife edge diffraction loss for this path as outlined above, and then add to it an excess loss factor L_{ex}, calculated from

$$L_{ex} = 11.7\alpha \sqrt{\frac{\pi r}{\lambda}} \; dB \qquad (14)$$

There is also a correction factor for roughness: if the object is, for example, a hill which is tree-covered rather than smooth at the top, the excess diffraction loss is said to be about 65% of that predicted in (14). In general, smoother objects produce greater diffraction losses.

Example 4. We revisit the scenario in Example 3, but let's suppose that we've now decided that the ridge blocking our path doesn't cut it as a knife edge (ouch!). From a plot of the profile, we estimate that D_s = 10 meters. As before, d_1 = 20 km, d_2 = 5 km and the height of the ridge is 100 meters. Dusting off our high school trigonometry, we can work out that a = 1.43°, or 0.025 radians. Now, plugging these numbers into (13), we get r = 188 meters. Then, with λ = 0.33 m, we can calculate the excess loss from (14):

$$L_{ex} = 11.7 \text{ x } 0.025 \text{ x } \sqrt{\frac{\pi \times 188}{0.33}} = 12.4 \; dB$$

So, summed with the knife edge loss calculated previously, we have an estimated total diffraction loss of 37.3 dB (assuming the ridge is "smooth" rather than "rough"). This is a lot, but you can easily imagine scenarios where the losses are much greater: just look at the direct dependence on the angle a in (14) and picture from Fig. 9 what happens when the obstacle is closer to one of the link endpoints. Amateurs doing weak signal work are accustomed to dealing with large path losses in non-LOS propagation, but such losses are usually intolerable in high-speed digital links.

Attenuation from Trees and Forests

Trees can be a significant source of path loss, and there are a number of variables involved, such as the specific type of tree, whether it is wet or dry, and in the case of deciduous trees, whether the leaves are present or not. Isolated trees are not usually a major problem, but a dense forest is another story. The attenuation depends on the distance the signal must penetrate through the forest, and it increases with frequency. According to a CCIR report [10], the attenuation is of the order of 0.05 dB/m at 200 MHz, 0.1 dB/m at 500 MHz, 0.2 dB/m at 1 GHz, 0.3 dB/m at 2 GHz and 0.4 dB/m at 3 GHz. At lower frequencies, the attenuation is somewhat lower for horizontal polarization than for vertical, but the difference disappears above about 1 GHz. This adds up to a *lot* of excess path loss if your signal must penetrate several hundred meters of forest! Fortunately, there is also significant propagation by diffraction over the treetops, especially if you can get your antennas up near treetop level or keep them a good distance from the edge of the forest, so all is not lost if you live near a. forest.

General Non-LOS Propagation Models

There are many more general models and empirical techniques for predicting non-LOS path losses, but the details are beyond the scope of this paper. Most of them are aimed at prediction of the paths between elevated base stations and mobile or portable stations near ground level, and they typically have restrictions on the frequency range and distances for which they are valid; thus they may be of limited usefulness in the planning of amateur high-speed digital links. Nevertheless, they are well worth studying to gain further insight into the nature of non-LOS propagation. The details are available in

many texts - Ref. [3] has a particularly good treatment. One crude, but useful, approximation will be mentioned here: the loss on many non-LOS paths in urban areas can be modeled quite well by a fourth-power distance law. In other words, we substitute d^4 for d^2 in equation (5). In equation (6), we can substitute $40\log(d)$ for the $20\log(d)$ term, which would correspond to the assumption of square-law distance loss for distances up to 1 km (or 1 mile, for the non-metric version of the equation), and fourth-law loss thereafter. This is probably an overly optimistic assumption for heavily built-up areas, but is at least a useful starting point.

The propagation losses on non-LOS paths can be discouragingly high, particularly in urban areas. Antenna height becomes a critical factor, and getting them up above rooftop heights will often spell the difference between success and failure. Due to the great variability of propagation in cluttered urban environments, accurate path loss predictions can be difficult. If a preliminary analysis of the path indicates that you are at least in the ballpark (say within 10 or 15 dB) of having a usable link, then it will generally be worthwhile to give it a try and hope to be pleasantly surprised (but be prepared to be disappointed!).

Software Tools for Propagation Prediction

Although there is no substitute for experience and acquiring a "feel" for radio propagation, computer programs can make the job of predicting radio link performance a lot easier. They are particularly handy for exploring "what if' scenarios with different paths, antenna heights, etc. Unfortunately, they also tend to cost money! If you're lucky, you may have access to one of the sophisticated prediction programs which includes the most complex propagation models, terrain databases, etc. If not, you can still find some free software utilities that will make it easier to do some of the calculations discussed above, such as knife edge diffraction losses. One very useful freeware program which was developed specifically for short-range VHF/UHF applications is **RFProp**, by Colin Seymour, G4NNA. Check Colin's Web page at http://www.users.dircon.co.uk/~netking/freesw.htm for more information and downloading instructions. This is a Windows (3.1, 95 or NT) program which can calculate path loss in free space and simple diffraction scenarios. In addition to calculating knife edge diffraction loss, it provides some correction factors for estimating the loss caused by more rounded objects, such as hills. It also allows changing the distance loss exponent from square-law to fourth-law (or anything else, for that matter) to simulate long paths with ground reflections or obstructed urban paths. There is also some provision for estimating the loss caused when the signals must penetrate buildings. The program has a graphical user interface in which the major path parameters can be entered and the result (in terms of receiver SNR margin) seen immediately. There is also a tabular output which lists the detailed results along with all of the assumed parameters.

Special Considerations for Digital Systems

We have previously looked at the effect of multipath on path loss. When reflections occur from objects which are very close to the direct path, then paths have very similar lengths and nearly the same time delay. Depending on the relative phase shifts of the paths, the signals traversing them at a given frequency can add constructively to provide a gain with respect to a single path, or destructively to provide a loss. On longer paths in particular, the effect is usually a loss. Since the path lengths are nearly equal, the loss occurs over a wide frequency range, producing a "flat" fade.

In many cases, however, reflections from objects well away from the direct path can give rise to significant multipath. The most common reflectors are buildings and other manmade structures, but many natural features can also be good reflectors. In such cases, the propagation delays of the paths from one end of the link to the other can differ considerably. The extent of this time spreading of the signal is commonly measured by a parameter known as the *delay spread* of the path. One consequence of having a larger delay spread is that the reinforcement and cancellation effects will now vary more rapidly with frequency. For example, suppose we have two paths with equal attenuation and which differ in length by 300 meters, corresponding to a delay difference of 1 μsec. In the frequency domain, this link will have deep nulls at intervals of 1 MHz, with maxima in between. With a narrowband system, you may be lucky and be operating at a frequency near a maximum, or you may be unlucky and be near a null, in which case you lose most of your signal (techniques such as space diversity reception may help, though). The path loss in this case is highly frequency-dependent. On the other hand, a wideband signal which is, say, several MHz wide, would be subject to only partial cancellation or *selective* fading. Depending on the nature of the signal and how information is encoded into it, it may be quite tolerant of having part of its energy notched out by the multipath channel. Tolerance of multipath-induced signal cancellation is one of the major benefits of spread spectrum transmission techniques.

Longer multipath delay spreads have another consequence where digital signals are concerned, however: overlap of received data symbols with adjacent symbols, known as *intersymbol interference* or ISI. Suppose we try to transmit a 1 Mbps data stream over the two-path multipath channel mentioned above. Assuming a modulation scheme with 1 μsec symbol length is used, then the signals arriving over the two paths will be offset by exactly one symbol period. Each received symbol arriving over the shorter path will be overlaid by a copy of the previous symbol from the longer path, making it impossible to decode. This problem can be solved by using an adaptive equalizer in the receiver, but this level of sophistication is not commonly found in amateur or WLAN modems (but it will certainly become more common as speeds continue to increase). Another way to attack this problem is to increase the symbol length while maintaining a high bit rate by using a multicarrier modulation scheme such as OFDM (Orthogonal Frequency Division Multiplex), but again, such techniques are seldom found in the wireless modem equipment available to hobbyists. For unequalized multipath channels, the delay spread must be much less than the symbol length, or the link performance will suffer greatly. The effect of multipath-induced IS1 is to establish an *irreducible error rate* - beyond a certain point, increasing transmitter power will cause no improvement in BER, since the BER vs E_b/N_0 curve has gone flat. A common rule of thumb prescribes that the multipath delay spread should be no more than about 10% of the symbol length. This will generally keep the irreducible error rate down to the order of 10^{-3} or less. Thus, in our two-path example above, a system running at 100K symbols/s or less may work satisfactorily. The actual raw BER requirements for a particular system will of course depend on the error-control coding technique used.

Delay spreads of several microseconds are not uncommon, especially in urban areas. Mountainous areas can produce much longer delay spreads, sometimes tens of microseconds. This spells big trouble for doing high-speed data transmission in these areas. The best way to mitigate multipath in these situations is to use highly directional antennas, preferably at both ends of the link. The higher the data rate, the more critical it becomes to use high-gain antennas. This is one advantage to going higher in frequency. The delay spread for a given link will usually not exhibit much frequency dependence - for example,

there will be similar amounts of multipath whether you operate at 450 MHz or 2.4 GHz, *if* you use the same antenna gain and type. However, you can get more directivity at the higher frequencies,* which often will result in significantly reduced multipath delay spread and hence lower BER. It may seem strange that high-speed WLAN products are often supplied with omnidirectional antennas which do nothing to combat multipath, but this is because the antennas are intended for indoor use. The attenuation provided by the building structure will usually cause a drastic reduction in the amplitude of reflections from outside the building, as well as from distant areas inside the building. Delay spreads therefore tend to be much smaller inside buildings - typically of the order of 0.1 μsec or less. However, as WLAN products with data rates of 10 Mbps and beyond are now appearing, even delay spreads of this magnitude are problematic and must be dealt with by such measures as equalizers, high-level modulation schemes and sectorized antennas.

Conclusions

Radio propagation is a vast topic, and we've only scratched the surface here. We haven't considered, for example, the interesting area of data transmission involving mobile stations - maybe next year! Hopefully, this paper has provided some insight into the problems and solutions associated with setting up digital links in the VHF to microwave spectrum. To sum up, here are a few guidelines and principles:

- Always strive for LOS conditions. Even with LOS, you must pay attention to details regarding variability of refractivity, Fresnel zone clearance and avoiding reflections from the ground and other surfaces. Non-LOS paths will often lead to disappointment unless they are very short, especially with the high-speed unlicenced WLAN devices. Their low ERP limits and high receive signal power requirements (due to large noise bandwidths, high noise figures and sometimes, significant modem implementation losses) leave little margin for higher-than-LOS path losses. Hams are not encumbered by the low ERP limits, but it can be very expensive to overcome excessive path losses with higher transmitter powers.

- Use as much antenna gain as is practical. It is always worthwhile to try both polarizations, but horizontal polarization will often be superior to vertical. It will generally provide less multipath in urban areas, and may provide lower path loss in some non-LOS situations (e.g., attenuation from trees at VHF and lower UHF). Also, interfering signals from pagers and the like tend to be vertically polarized, so using the opposite polarization can often provide some protection from them.

- There are advantages to going higher in frequency, into the microwave bands, due to the higher antenna gains which can be achieved. The tighter focusing of energy which can be achieved may result in lower overall path loss on LOS paths (providing that you can keep the feedline losses under control), and less multipath. Higher frequencies also have smaller Fresnel zones, and thus require less clearance over obstacles to avoid diffraction losses. And, of course, the higher bands have more bandwidth available for high-speed data, and less probability of interference. However, the advantage may be lost in non-LOS situations, since diffraction losses, and attenuation from natural objects such as trees, increase with frequency.

Radio propagation is seldom 100% predictable, and one should never hesitate to experiment. It's very useful, though, to be equipped with enough knowledge to know what techniques to try, and when there is little probability of success. This paper was intended to help fill some gaps in that knowledge. Good luck with *your* radio links!

Acknowledgements

The author gratefully acknowledges the work of his daughter Kelly (http://hydra.carleton.ca/~klm) in producing the figures for this paper. WaveLAN is a registered trademark of Lucent Technologies, Inc.

References

[1] *ARRL UHF/Microwave Experimenter's Manual* (American Radio Relay League, 1990).

[2] Hall, M.P.M., Barclay, L.W. and Hewitt, M.T. (Eds.), *Propagation of Radiowaves* (Institution of Electrical Engineers, 1996).

[3] Parsons, J.D., *The Mobile Radio Propagation Channel* (Wiley & Sons, 1992).

[4] Doble, J., *Introduction to Radio Propagation for Fixed and Mobile Communications* (Artech House, 1996).

[5] Bertoni, H.L., Honcharenko, W., Maciel, L.R. and Xia, H.H., "UHF Propagation Prediction for Wireless Personal Communications", *Proceedings of the IEEE*, Vol. 82, No. 9, September 1994, pp. 1333-1359.

[6] Andersen, J.B., Rappaport, T.S. and Yoshida, S., "Propagation Measurements and Models for Wireless Communications Channels", *IEEE Communications Magazine*, January 1995, pp. 42-49

[7] Freeman, R.L., *Radio System Design for Telecommunications* (Wiley & Sons, 1987).

[8] Lee, W.C.Y., *Mobile Communications Design Fundamentals*, Second Edition (Wiley & Sons, 1993).

[9] CCIR (now ITU-R) Report 567-4, "Propagation data and prediction methods for the terrestrial land mobile service using the frequency range 30 MHz to 3 GHz" (International Telecommunication Union, Geneva, 1990).

[10] CCIR Report 1145, "Propagation over irregular terrain with and without vegetation" (International Telecommunication Union, Geneva, 1990).

Appendix

Cable Type	144 MHz	220 MHz	450 MHz	915 MHz	1.2 GHz	2.4 GHz	5.8 GHz
RG58	6.2 (20.3)	7.4 (24.3)	10.6 (34.8)	16.5 (54.1)	21.1 (69.2)	32.2 (105.6)	51.6 (169.2)
RG-8X	4.7 (15.4)	6.0 (19.7)	8.6 (28.2)	12.8 (42.0)	15.9 (52.8)	23.1 (75.8)	40.9 (134.2)
LMR-240	3.0 (9.8)	3.7 (12.1)	5.3 (17.4)	7.6 (24.9)	9.2 (30.2)	12.9 (42.3)	20.4 (66.9)
RG-213/214	2.8 (9.2)	3.5 (11.5)	5.2 (17.1)	8.0 (26 2)	10.1 (33.1)	15.2 (49.9)	28.6 (93.8)
9913	1.6 (5.2)	1.9 (6.2)	2.8 (9.2)	4.2' (13.8)	5.2 (17.1)	7.7 (25.3)	13.8 (45.3)
LMR-400	1.5 (4.9)	1.8 (5.9)	2.7 (8.9)	3.9 (12.8)	4.8 (15.7)	6.8 (22.3)	10.8 (35.4)
3/8" LDF	1.3 (4.3)	1.6 (5.2)	2.3 (7.5)	3.4 (11.2)	4.2 (13.8)	5.9 (19.4)	8.1 (26.6)
LMR-600	0.96 (3.1)	1.2 (3.9)	1.7 (5.6)	2.5 (8.2)	3.1 (10.2)	4.4 (14.4)	7.3 (23.9)
1/2" LDF	0.85 (2.8)	1.1 (3.6)	1.5 (4.9)	2.2 (7.2)	2.7 (8.9)	3.9 (12.8)	6.6 (21.6)
7/8" LDF	0.46 (1.5)	0.56 (2.1)	0.83 (2.7)	1.2 (3.9)	1.5 (4.9)	2.3 (7.5)	3.8 (12.5)
1 1/4" LDF	0.34 (1.1)	0.42 (1.4)	0.62 (2.0)	0.91 (3.0)	1.1 (3.6)	1.7 (5.6)	2.8 (9.2)
1 5/8" LDF	0.28 (0.92)	0.35 (1.1)	0.52 (1.7)	0.77 (2.5)	0.96 (3.1)	1.4 (4.6)	2.5 (8.2)

Table 1

Attenuation of Various Transmission Lines in Amateur and ISM Bands in dB/100 ft (dB/100 m)

Notes

Attenuation data based on figures from the 'Communications Coax Selection Guide" from Times Microwave Systems (http://www.timesmicrowave.com/products/commercial/selectguide/atten/) and other sources.

The LMR series is manufactured by Times Microwave. 9913 is manufactured by Belden Corp. RG-series cables are manufactured by Belden and many others. The LDF series are foam dielectric, solid corrugated outer conductor cables, best known by the brand name HELIAX (@Andrew Corp.).

Software Radio Technology Overview And Recent Progress
Digital Communications Conference

Joseph Mitola III
The MITRE Corporation, McLean, VA 22102
(jmitola@mitre.org)

Abstract

This paper summarizes software radio technology emphasizing recent progress, including the first software radio workshop of the European Community and progress of the MMITS (open architecture software radio) Forum.. The software radio is an emerging technology for rapidly building flexible, modular, multiband multimode radio systems. It allows one to create radio infrastructure that can be programmed for new standards and dynamically updated with new software personalities. These personalities include air interfaces that may be downloaded to software radios "over the air", reducing the need to purchase new hardware for new services. The technology has been proven in the field, but there are technical, economic and institutional challenges remaining before the benefits of this technology are fully available at low cost. This paper highlights key technical challenges and opportunities.
Keywords: Software Radio. Digital RF. DSP. Architecture

1.0 Introduction

With a *true* ***software*** *radio,* the user can upload new air interfaces and protocol personalities as software radio "applets". This flexibility is a key reason for the rising significance of software radio technology: the software radio's multiband multimode waveforms are software-defined and this allows users and service providers (military, civil and commercial alike) next-generation flexibility in the design and evolution of wireless networks, infrastructure and services. Although the technology has been demonstrated in military radio research programs such as the Defense Advanced Research Projects Agency's (DARPA's) SPEAKeasy program, most of the potential has not yet been realized in the commercial sector. But since the technology's costs are declining with improvements in Digital Signal Processing (DSP) technology, it bears watching for applications in amateur radio. In fact, an early progenitor of the software radio is Standard Marine AB's HF multimode radio[1]. Although this product had only a 28 kilo sample per second Analog to Digital Converter (ADC), it implemented half a dozen HF radio modes in software. With contemporary technology, the digitally accessible bandwidths are now about 12 MHz with an 80 dB full-band dynamic range or 1 MHz with a 90 dB dynamic range. Such wideband IF access provides new opportunities for HF, line of sight and satellite amateur radio communications products including new air interfaces.

1.1 Disclaimer

The author is employed by The MITRE Corporation, a not-for-profit company that operates Federally Funded Research and Development Centers (FFRDCs) for the US Department of Defense (DoD). At present, he is on loan to the DoD as director of modeling and simulation of telecommunications systems. Remarks presented here are his alone and are not necessarily the views or policy of The MITRE Corporation or the US Government.

1.2 Perspective

Software radio technology has its roots in the Electronic Warfare (EW) programs of the 1980's which accessed the entire electromagnetic spectrum to monitor threats and to inject jamming signals. Thus, the military radio and EW technology is available to build wideband antennas; wideband radio frequency (RF) modules; high sampling rate Analog to Digital Converters (ADCs) and DACs; and the high performance digital signal processors needed for software radios - but at what cost? Commercial applications, especially amateur radio applications, and large scale civilian radio markets are very cost sensitive. So the focus of this paper is on technologies for improved cost/benefit and broader applications of software radio technology.

The software radio was identified as a key enabling technology of the "future-proof' infrastructure needed by wireless service providers in BellSouth's December 1995 Request for Information for the Software-Defined Radio [2]. The US Federal Aviation Administration (FAA) also requested industrial participation in inserting this technology into future avionics and ground based radio infrastructure [3]. A cross-section of over 100 government and commercial players created the Modular Multifunction Information Transfer System (MMITS) Forum in March of 1996 to promulgate the benefits of open architecture plug-and-play software radio and digital communications to an expanding marketplace [4]. MMITS has included Alcatel (France), Nokia and Ericsson (Sweden), Orange (UK), Samsung (Korea), and Raphael (Israel), and Computer Aided Design (CAD) software vendors, among others, including numerous US commercial and military radio product suppliers. In addition, the first European Workshop on Software Radios, held in Brussels on May 29 brought together over 160 European telecommunications systems professionals to deliberate the European approach to this technology. With so many players across the global landscape engaged in an emerging technology it is essential to first agree on a few definitions.

Table 1 Software Radio Functional Definition

General Properties	Universal air interfaces (source coding, channel coding, error control and protocols), regardless of **multi-technology (FDMA,** TDMA, CDMA or hybrids), **multi-band and multi-standard environments.**
Services*	Seamless internetworking of **AM, FM,** cellular **(analog,** TDMA, CDMA), PCS, mobile data and paging; **seamless bridging of multiple bands and modes**
Standards*	**HF ALE, VHF/UHF** voice/data; **privacy;** GSM, PCS and **Frequency Hop (FH).**
Technical Flexibility*	**Flexible RF, Channel,** Time Slot, Power, Bit rate, Equalization, **Channel Coding** and Error Correction.
Supports Advances	Adaptive networks, **transparent bridging,** innovative signaling and improved quality. Over the air downloading of radio personalities
Growth Path	Velcro radio (multiple hardware personalities) -> **DSP-enabled radio** -> **Multi-personality radio -> Variable personality software radio.**

Lists are illustrative, not exhaustive **[SPEAKeasy Demonstrates the Capabilities Shown In Bold]**

2. Software Radio Definitions

2.1 Functional Definition

The service providers - military, civilian and commercial - are most interested in the functional capabilities and cost of the software radio and less

interested in the technical details. The functional definition of Table 1 captures the significant functional dimensions of the software radio. SPEAKeasy, the first widely published military software radio, demonstrated the functions shown in bold in the table [5, 6]. The BellSouth software-defined radio growth path envisions a shift from multiple chip sets, one for each air interface (the "velcro" radio) to the Digital Signal Processing (DSP) - enabled radio. DSP includes not just DSP chips such as Texas Instruments' TMS320 and the ADSP SHARC, but also Field Programmable Gate Arrays (FPGAs) and general purpose processors such as Intel's Pentium/ MMX. As technology evolves software will also control RF analog processing through programmable Micro Electra-Mechanical Devices (MEMS).

2.2 Architecture Definition

The architecture of the software radio is described in detail elsewhere [7, 8]. Fundamental to the definition of the software radio is the use of wideband ADCs, DACs and high performance DSPs to define waveforms in software at Intermediate Frequencies (IF), improving on our recent ability to define waveforms at baseband. Soon technology will emerge for defining waveforms digitally at Radio Frequencies (RF). The other fundamental concept is to host all the software (DSP and otherwise) on general purpose programmable processors. All aspects of the air interface including the channel waveforms would then be defined in software and implemented in real time through isochronous software and/or firmware (versus dedicated digital hardware as in "digital" radios). These fundamentals cannot be achieved without wideband antennas[1], wideband RF amplifiers and RF distribution and (for now at least) wideband IF ADCs and DACs.

The software radio functional architecture block diagram of Figure 1 shows how the wideband ADC provides simultaneous access to a large population of channels, e.g. for subscribers in Personal Communications System (PCS) base stations. Software-based isochronous signal streams support the air interface. The environment characterization near-real-time stream provides parallel access to all the channels, e.g. for adaptive channel assignment algorithms. software radios also have a role in R&D where advanced services like joint source-channel coded variable data rate air interfaces are defined in software and targeted for software, firmware and/or hardware implementations via C and VHDL.

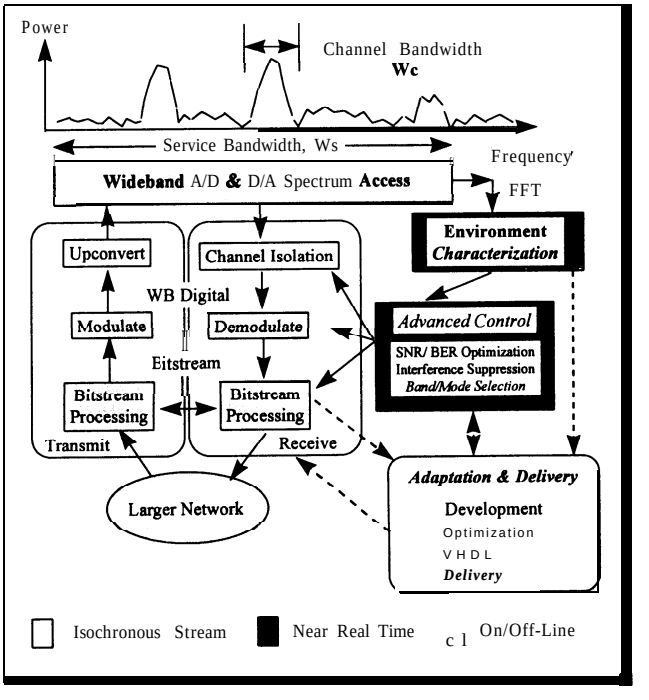


Figure 1 Software Radio Architecture

2.3 The Phase Space Definition

Digital radios and software radios may also be defined in the "phase space" of Figure 2. [Phase

[1] The SPEAKeasy I military software radio, for example, used three antenna bands between 2 MHz and 2 GHz, approximately 2-30, 30-300 and **300-2000** MHz.

spaces are used by physicists to represent changes in the states of matter as a function of key external parameters like temperature and pressure; we borrow the terminology to reflect the states of radio devices with respect to the key parameters of maximum frequency accessed digitally and degree of programmability.] The "ideal" software radio accesses RF directly via super-wideband ADC/DACs and accomplishes all processing using general purpose computer chips (see the circle marked X in the upper right corner of the figure).

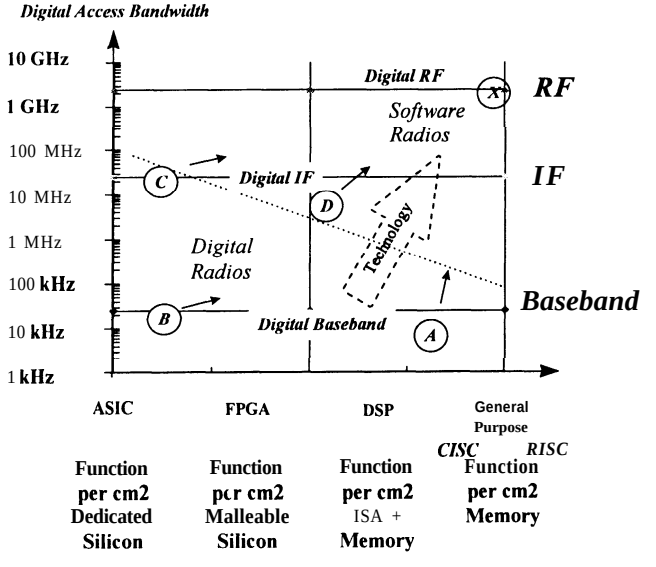


Figure 2 Software Radio Phase Space

Such software radios cannot be built economically yet. The pure digital radio, on the other hand, accomplishes most functions in Applications-Specific Integrated Circuits (ASICs) (circle C in the figure). The software radio maximizes flexibility and therefore truly future-proofs the infrastructure against new standards. The digital radio, conversely, maximizes hardware efficiency and therefore minimizes the size, weight and battery drain critical for handset applications.

Neither approach is a panacea: the key question is the degree of programmability required for the intended market. Contemporary radio designs therefore vary across the dotted line in the phase space that represents the technology frontier, comprising a mix of ASIC, FPGA, DSP and general purpose processors using ADCs and DACs at baseband or IF. The circle marked A in the figure shows the design point for the commercially successful Standard Marine HF amateur radio product [1], for example. Advancing microelectronics technology moves all implementations upward and to the right over time.

Handsets favor ASICs and/or FPGAs with chip level integration. Emerging cellular base stations, on the other hand, favor larger granularity hardware modules. Some use block up and down conversion with 12.5 to 25 MHz bandwidth ADCs/DACs (30 to 70 M samples per second) to accommodate 1 00-plus subscribers. This software radio approach reduces the hardware complexity of a cell site from several racks of discrete single-channel radios to one or two shelves of open architecture PCI, VME or other low cost DSP hardware. Military radio, avionics and amateur radio markets fall between handsets and base stations. Typically two dozen channel waveforms ("modes") are needed from HF to SHF for military radio interoperability, but no more than a half dozen modes are in use at any one time per platform (e.g. tank, aircraft, etc.). Amateur radio applications might employ two or three simultaneous modes out of a couple of dozen available. Simultaneous voice, automatic digital QSL, packet satellite and monitoring favorite channels could all be supported in an amateur software radio.

2.4 The Vector Space Definition

Two years ago, nobody had a "software radio" and now almost every radio vendor on the planet claims to have one. This confuses buyers, program managers and investors. We may clarify things somewhat by defining the degree to which a product meets four key technical and economic criteria that bridge the gap from technologies to applications. The criteria are:

1. The number of air interface channels simultaneously supported (N),
2. Programmable Digital Access (PDA),
3. Hardware Modularity **(HM),** and
4. Software Flexibility and Affordability (**SFA**)

Table 2 defines these criteria. It is useful to group (N), the number of air interface channels, into four types: single channel; dual channel; multiple channel (i.e. less than 6); and full access (i.e. the full number of subscribers in an allocated RF band) to define four levels of N. Multiple channel nodes are typical of military, civil and amateur radio building blocks while the full band modes are typical of cell site infrastructure.

Table 2 Definitions of Four Key Software Radio Dimensions (N, PDA, HM, SFA)

N: Number of Channels:
n: 1,2 or a few (<6) simultaneous air interface channels;
NN: The full number of subscribers in the RF band.
PDA: Programmable Digital Access: None (0), Baseband (1), IF (2), RF (3);
Baseband bandwidth is defined by single subscriber service (e.g. voice, data modem, video)
IF is defined as that bandwidth which simultaneously supports all NN subscribers in the allocated
RF service band (e.g. 12.5 MHz analog FDMA)
HM: Hardware Modularity:
None (0), Receiver/Exciter/INFOSEC/Network Modules (1), COTS DSP Modules (2),
Second Level Modules (ADCs, FPGAs, Receiver Chips, etc.) (3);
SFA: Software Flexibility and Affordability:
No Air-interface-defining software (0), Single-supplier software (1),
Multiple supplier but single host platform (2), Multiple supplier multi-platform software (3).

The level of Programmable Digital Access **(PDA)** is the level of the conversion to digital at which the radio is functionally programmable in the software radio phase space. The types are: none (totally analog or fixed function digital radio); baseband programmability; IF programmability; and RF programmability. A privacy mode that hops over 2 MHz but that cannot be programmed for any other waveform may have a wideband IF, but it would not have a ***programmable digital access at*** *IF,* which is the criteria specified in this definition.

The Hardware Modularity **(HM)** criterion recognizes the differences in upgrade path between relatively coarse grain (possibly programmable) modules such as receivers and exciters; versus other types of coarse grain modules (e.g. COTS ADC and DSP boards); versus finer grain modules such as FPGA, ADC and DSP chips. The hardware modularity value is not prejudicial: The key is to explicitly decide what type of modularity is called for by the life cycle of the application and to match that type in the implementation. Amateur radio applications favor PC board level modules.

The Software Modularity (SFA) dimension characterizes the service provider's ability to buy plug and play software modules based on the vitality of the marketplace. Software that runs on just one platform and is available from only the original manufacturer tends to box the user into single-source (sometimes very expensive) maintenance. If the functionality of the unit will not change over its life cycle, then this may be a perfectly

acceptable path. This would be a rare occurrence in today's fast changing marketplace. Software that runs on many platforms (e.g. JAVA) and is available from multiple vendors generally gives the buyer a better software product with more flexibility and at a lower cost over the life cycle.

We may think of these criteria as a feature space, yielding a characteristic vector for a given radio product:

Radio X: (Nx, DAx, HMx, SFAx)

Each vector element varies between 0 and 3 per the assigned type, so the vectors range from (0000), the unprogrammable analog radio to (3333), the totally programmable software radio. Since there are four levels of capability for each of these four features, there are a total of 256 points in this feature space. These 256 points cluster into four groups that may be called capability levels.

2.5 Software Radio Capability Levels

The clusters of software radio feature vectors consist of the four aggregate "software radio capability levels" shown in Table 3. Radios at level zero have fixed functionality and cannot be programmed by the user. Level one radios have programmable digital basebands while level two radios have programmable digital IF's. Level three, programmable digital RF, is the ideal software radio which with today's technology is unaffordable. Level zero is not necessarily bad or low-technology. Level zero includes digital baseband dedicated function chip sets that can have very high levels of' device technology (e.g. GSM handsets). For example, simple dual mode handsets need no air interface programmability, so non-programmable ASIC chips are generally most cost effective (e.g. for GSM/ CDMA handsets).

Table 3 Definition of Significant Software Radio Capability Levels

Software Radio Level	Characteristics [Examples with level rating (N, PDA, HM, SFA)]
Zero **Fixed Functionality**	**Analog Radios** [Walkie Talkie (1 ,0,0,0), FM FDM (NN,0,0,0)] **Digital Readout** [Pager (1, I ,0,0) Direct Conversion Handset (1, I ,0,0)]
One **Programmable Digital (Baseband) Radios**	**"Narrowband" reprogrammable, modular (*,1 ,>0,>0)** [Contemporary programmable digital radios Closed architecture (1 -n, 1 ,0, 1), modular (1 -n, 1,1,1) "Open Architecture " (I -n, 1,2,1-2), Goal (1 -n, 1,1-3,3)]
Two **Programmable Digital-IF Radios**	**"Narrowband" programmable , some wideband hardware (*,2,>0,>0)** [SPEAKeasy Class (1 -n,2,2/3,1 or 2?) A Few New Cell Site Products (NN,2,1/2,1)
Three **The Software Radio**	**Programmable Digital RF ["Ideal" Software Radio (NN,3,1-3,3)]** - Not affordable yet, but useful as a migration challenge

The aggregate software radio level does indicate the degree of air-interface programmability by the user. Contemporary programmable digital radios meet level one criteria, with narrowband programmability through baseband digital signal processing. From software radio feature vectors, it is easy to see the differences among level one implementations with closed architectures and no

real hardware modularity or growth path (1-n, 1 ,0, 1), modular hardware (l-n, 1 ,1 , 1), some degree of modular hardware and software for nominally "open architecture" (1 -n, 1,2,1-2); and the highly modular widely supported open architecture goal (1-n,1,1-3,3).

Contemporary programmable digital radios seek to achieve level two, programmable digital IF, in the near to mid- term. The US SPEAKeasy program, for example, reaches toward level two with medium bandwidth ADCs and DACs and nearly 1 GFLOP of DSP. The need for simultaneous channels and flexibility in the field provided by software radio technology must be balanced against other competing demands of the market segment as illustrated in Figure 3. These market segment drivers create a set of engineering, design and regulatory challenges and opportunities.

Market Segment	Simultaneous Channels/Modes	Architecture Drivers	Standards
Handset	1-2+GPS	Mfg Volume "Velcro"	Chip Level Interfaces
Manpack/ Avionics	4-20	Size, Weight, Power	PC1 PCMCIA
Law Enforcement	20-1 oo+	Cost	?
PBX, WLL	20-1 oo+	Call Quality	?
Base Station/ & Mobile Bases	>100	Future-Proof DSP Leverage	VME-like +Wideband Bus

Figure 3 Market Segment Drivers

3 Key Challenges

In order to move as an industry from level one to level two of software radio flexibility and affordability, we need higher quality RF access and better partitioning of the systems and software for modular plug-and-play services and support as highlighted in Figure 4. None of these challenges has an easy, inexpensive and near-term solution, but all are being addressed by technology investments currently under way.

- **High Quality RF Access**
  - Increased Useful Wideband Dynamic Range with Improved Noise Immunity
  - Lower Cost of Broad RF Access
  - Reduced Cosite Interference
- **Partitioning for Plug-and-Play and Reuse**
  - MMITS "API" Approach
  - Real Time CORBA
  - Z. 100 Communications Language

Figure 4 Key Software Radio Challenges

3.1 High Quality, Low Cost RF Access

Software radios depend on high quality low cost access to broad ranges of the RF spectrum. Although the ADC plays a key role, the useful dynamic range is defined by two-tone spurious-free dynamic range (SFDR) established by the multiplicative effects of RF conversion, ADC and subsequent digital filtering. As noise and interference from the environment aliases into the ADC passband, it reduces the sensitivity of the overall system. The key challenge in the short term is to reduce such noise and interference through technical advances in antennas, RF analog filters, ADCs and digital filters.

The costs of RF access are driven by mechanical RF structures (antennas, waveguide, coax and other "plumbing") which are large and expensive because of the relatively large number of discrete parts and the high labor content of assembly and installation. The recurring costs of software radio nodes may include upwards of 60% for wideband antennas, RF distribution, RF conversion and IF processing but as little as 10% for Commercial Off The Shelf (COTS) DSP hardware. As digital hardware moves towards the antenna, it brings the advantages of rapidly advancing microelectronics technology including reduced production costs. There are standard RF packages, waveguide and connectors but many RF assemblies are virtually

hand crafted on the "production" line. The matching of voltage standing wave ratios, trimming of capacitance, and other touch labor increases costs. Digital technology, on the other hand, reduces or eliminates most such manufacturing steps. The key challenge for the long term is to move more from analog RF to digital RF.

Finally, cosite interference is mostly our own fault. In the early days of radio, the size and expense of the radio equipment made Frequency Domain Duplexing (FDD) economically infeasible for most applications. As a result, many bands have a legacy of Time Domain Duplexed (TDD) air interfaces in the HF, VHF/ UHF, satcorn and other bands. Any software radio that attempts to service all the users in a TDD band with a single (low cost) RF/IF/ADC channel has the problem of "screaming in its own ears" as multiple TDD signals are transmitted and received at the same time. There are a few technical approaches to mitigate such interference, but mitigation beyond 20 to 30 dB requires a research breakthrough. One could dramatically reduce cosite interference in the far term through a spectrum use paradigm shift from TDD to FDD.

3.2 Plug-and-Play and Reuse

Plug-and-pIay radio would bring the benefits of the open architecture desktop to most amateur radio applications. Reuse can be accomplished on at least two levels: software reuse and waveform reuse. Both plug-and-play and reuse depend on a workable partitioning of the software into modular functions with clearly defined and broadly accepted interfaces. The MMITS forum is pursuing a partitioning based on an Applications Programming Interface (API). The Object Management Group (OMG) recently requested proposals for real-time multimedia support for its Common Object Request Broker Architecture (CORBA)[9]. Multimedia support should add isochronous channels to a growing repertoire of CORBA capabilities relevant to plug-and-play and reuse in software radios. CORBA has little domain-specific (radio voice and data) representation ability. This is both a strength and a weakness. The strength is that CORBA is widely applicable. The weakness is that one must define one's own radio domain representations. As a result, the representation of radio engineering interfaces among software objects such as state machines must be found elsewhere. The Z.100 Recommendations of the ITU-T [10] describe the Specification and Description Language (SDL) that provides a rich set of expressions of telecommunications behavior including call processing; maintenance and fault treatment; system control; data communications; and some telecommunications services. SDL is compilable with a wide base of European users. The key challenge is the integration of CORBA and SDL in an API such as that contemplated by the MMITS forum for commercial plug-and-play software radios.

4.0 Technology-Enabled Opportunities

These are significant challenges, but many have related enabling technologies to which we can look for continued progress through investments focused as suggested in Table 4.

Table 4 Enabling Technologies

Technology	Representative COTS Performance	Investment Focus
Multiband Multibeam Antennas	Decade (e.g. 0.4 to 3 GHz)	Gain, physical size, alignment, cost
Wideband RF	Octave to decade	Automatic mechanical tuning, MEMS
Wideband ADC	70 MHz x 12 bits	Dynamic Range x Bandwidth
HTSC Filters and Amplifiers	30-40 dB More Out of Band Rejection	Product Integration
High Performance interconnect	140 to 1000 Mbytes per second	User-accessible throughput
Digital Signal Processors	25 MFLOPS to 2000 MIPS	Throughput, power consumption
Real Time Object Software	COTS Radio / Telephony Functions Real Time CORBA for Multimedia ITU-T Z. 100 SDL Products	Quantified Real-Time Performance Object Oriented packages Integration of APIs

4.1 Sensitivity and Dynamic Range

The Commercial Off The Shelf (COTS) multiband multibeam antennas, wideband RF and wideband ADC products of Table 4 may be enhanced for software radio applications through the technology investment focus areas shown. High Temperature Super-Conductive (HTSC) filters now entering the market suppress adjacent channel noise and interference by 30 to 40 dB [11]. Otherwise, this interference would alias into a software radio's wideband programmable IF. Conductus, Inc. (Sunnyvale, CA) and Illinois Superconductor Corporation (Mt. Prospect, IL) are among several companies to offer HTSC filters and/or low noise amplifiers. Ericsson's presentation at the European Conference[12] emphasized the relaxation of test requirements for GSM so that today's ADC and filter technology could be applied immediately for multimode base stations.

In the Proceedings of the GaAs IC Symposium [13], Walden reported ADCs with 6 bits of resolution at 6 GHz, allowing one to sample a 2.5 GHz RF waveform above the Nyquist sampling criterion. The two tone spurious free dynamic range (SFDR) of this ADC may approach 30 dB, but most radio applications require 70 to 90 dB or more dynamic range. The ADC seems inadequate until oversampling is considered. The oversampling gain in dynamic range (DNR) is approximately the ratio of the sampling rate to the Nyquist rate as shown in Table 5, provided sampling clocks have the aperture uncertainty and stability necessary for coherent integration. Vetterli has described projection filters that optimize dynamic range recovery [14]. The preservation of SFDR from GHz sampling rates to narrowband modulated channels has not yet been achieved in practice but is an active research area. Current research emphasizes higher speed and dynamic range ADCs and DACs [15] with demultiplexers to reduce data handling clock speeds.

4.2 Digital RF'

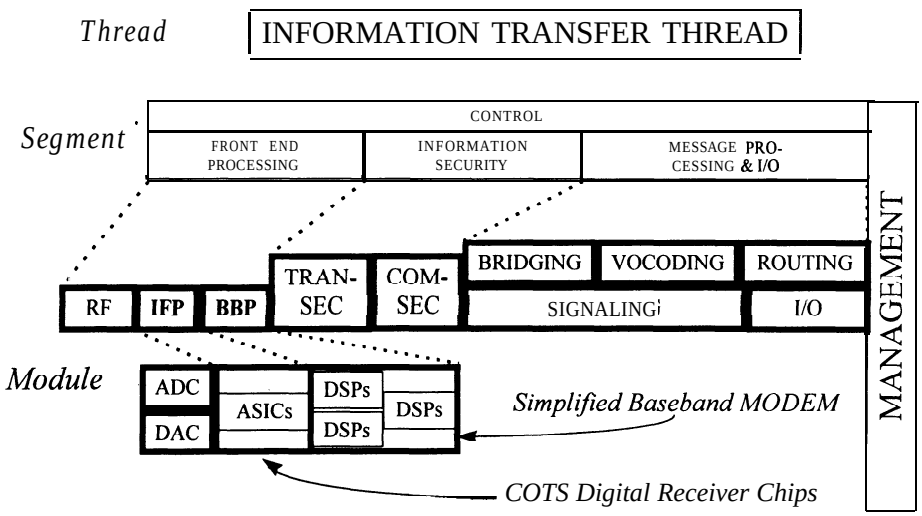
Table 5 Dynamic Range Improves With Oversampling

(Over)Sampling Rate MHz	ADC's SFDR	Cell Site Bandwidth MHz	Gain In DNR	Cell DNR	Subscriber Bandwidth kHz	'Subscriber DNR
6000	30	25	21	51	30	80

5.0 Plug-and-Play and Reuse

5.1 The MMITS API Concept

The MMITS Forum has begun to define open architecture standards for plug and play communications embracing digital and software radios, offering the "Information Transfer Thread," a refinement of which is shown in Figure 5.



IFP = IF Processing; BBP = Baseband Processing

Figure 5 Refined Information Transfer Thread Reference Model

The heavy lines in the figure represent Applications Program Interfaces (APIs) mediated by a common bus (e.g. the red or black bus). The dotted lines link successive levels of expansion of the hierarchical decomposition of the Information Transfer (IT) Thread. At the top level, the IT Thread consists of a front-end; an information security (INFOSEC) element; and a back-end for message processing ("inter-networking"), and input/output to user(s) and possibly to wireline interfaces. The functional interfaces at this level provide a reference model for building radio applications via a standard API.

5.2 Real Time Software Objects

Non-recurring engineering often includes greater than 50% software development costs. The software is generally on the critical path with system integration and test as custom software, COTS hardware and operating systems and signal processing libraries are fused into a viable system. Object oriented languages and design have entered the mainstream of real-time applications [16,171, but the impact of Common Object Request Broker Architectures (CORBA and related standards[18]) has not yet reached the front end of the architecture. That is, a user or systems developer cannot yet substitute one RF modem software component for another without relatively labor-intensive customized coding and system tuning.

This is in part due to the lack of standard applications level software to software APIs and in part due to a lack of quantification of requirements for memory, buffer space and processing resources. Orange Communications, Nokia and others at the

First European Conference [12] emphasized the need to develop such API's. It is difficult to tell if one *could* meet throughput, response time and other critical requirements when attempting to reuse code from one's own libraries or from a third party. Quantified software objects specify applications layer interfaces and related memory and processing resource requirements for real-time performance. As such, they reduce the time required and risk of buying software that will not work effectively in one's PC or laptop environment. A reusable object could use CORBA interfaces for plug and play, but this does not guarantee performance. In fact, the CORBA overhead detracts from throughput and response time. So a quantified object characterizes the processing demand that must be supported for real-time performance.

6.0 Conclusions

As digital radios make the transition to software radios, industry is striving for economic efficiency through the adoption of open architecture standards and through technology insertion. Workshops such as the first European Workshop on Software Radios certainly enhance that process through stimulating critical technical and business dialogs. By adopting standard terminology to describe different kinds of software radios, we can enhance our ability to communicate technically and with customers and investors. We also need a accurate communications between buyers and sellers of amateur radio digital hardware and software products. The software radio feature space is offered as a useful characterization of product technology and robustness. As suppliers offer more software radio products, working in a standard API such as the MMITS API and quantified so that procurement risk is low, we will see much broader market acceptance of the software radio.

References

1 Standard Marine AB Product Description, 1991

2 The Software Defined Radio, BellSouth, Dec 1995

3 Broad Agency Announcement, The Federal Aviation Administration, Commerce Business Daily (12 Sept 96)

4 MMITS Forum Charter (Internet www.rl.af.mil/MMITS; April 95)

5 Lackey and Upmal, "SPEAKeasy: The Military Software Radio," IEEE Communications Magazine (IEEE: NY), May 95

6 Fette, B. "SPEAKeasy II Program Overview", Minutes of the MMITS Forum (Reference 3) June and March 96.

7 Mitola, "The Software Radio: Architecture and Prognosis, IEEE National Telesystems Conference 92 (IEEE, NY)

8 Mitola, "The Software Radio Architecture", IEEE Communications Magazine (IEEE, NY; May 95).

9 Object Oriented Technology, IEEE Communications Magazine (IEEE, NY), Feb 97.

10 CCITT Specification and Description Language, Recommendation Z. 100 (ITU, Geneva, Switzerland), 1993.

11 Hogan, "Superconductor products poised for market", *The Industrial Physicist* (Woodbury, NY), March 1997.

12 First European Conference on Software Radios, European Commission, Brussels, May 29 1997

13 Walden, "ADC Integrated Circuits", Proceedings of the IEEE GaAs IC Symposium (IEEE Press, NY) Dec 95.

14 Thao&Vetterli, "Optimal MSE Signal Recovery in Over-sampled A/D Conversion Using Convexity", IC-ASSP-92.

15 Lemnios, Zachary, DARPA ADC Program Review, (DARPA, Arlington, VA), March 96.

16 Ellison, Karen S., Developing Real-Time Embedded Software-In A Market-Driven Company, (Wiley, NY) 96

17 Ellis, J. Objectifying Real-Time Systems, (SIGs Books, NY) 1995.

18 The Common Object Request Broker: Architecture and Specification, OMG 9 1.12.1 (DEC, Maynard MA) 1993

PerlAPRS

An Automated Control Application for APRS Networks

Richard Parry, P.E., W9IF
rparry@qualcomm.com
http://people.qualcomm.com/rparry

ABSTRACT

PerlAPRS is an application which can monitor both local TNC received APRS packets and remote Internet APRS packets and perform an automated action based on criteria specified by the user. The criteria that PerlAPRS uses is the callsign of the station and its location specified as a Maidenhead Grid Square. Other requirements specified by the user increase functionality of the program in real world applications. The actions executed can be written in any language, but UNIX style shell scripts are ideally suited for this purpose. Scripts can be developed to perform functions such as automatic notification via email as well as logging. PerlAPRS is freely distributed under the GNU licensing agreement.

KEYWORDS

Packet Radio, APRS, Linux, UNIX, Perl

INTRODUCTION

The Automatic Position Reporting System' is one of the most popular facets of amateur radio today. It is a marriage of several cutting edge technologies including the Global Positioning System (GPS), amateur packet radio, and the global Internet. It incorporates satellite technology, wireless networks, and both analog and digital communication. The applications to support the APRS protocol are also sophisticated. They provide the user with an easy to use interface into the APRS world. Software such as MacAPRS for the Apple Macintosh, WinAPRS for Windows 95/NT, and APRSdos for DOS machines provide powerful and elegant solutions. With this software and support system, it is possible to display any APRS network. Other support software for APRS includes the work of Steve Dimise, K4HG, who extended the concept for the promulgation of APRS packets to the Internet with javAPRS. In addition, Steve Boyle, KD6WXD; and Dale Heatherington, WA4DSY, developed APRS servers for the Internet which allow users to remotely connect to the server and examine remote APRS networks.

These programs provide flexibility, functionality, and a highly visual means for tracking APRS activity. However, they are passive in that they provide predominately monitoring functionality. They do not provide the ability to control. For example, if you wish to know when an APRS tracker escorting marathon runners reaches a specific location, you need more than monitoring capability, you need control functionality. It is this ability that PerlAPRS provides.

---

[1] The APRS formats are provided for use in the amateur radio service. Hams are encouraged to apply the APRS formats in the transmission of position, weather, and status packets. However, APRS is a registered trademark of Bob Bruninga who reserves the ownership of these protocols for exclusive commercial application and for all reception and plotting applications. Other software engineers desiring to include APRS protocols in their software for sale within or outside of the amateur community will require a license from him.

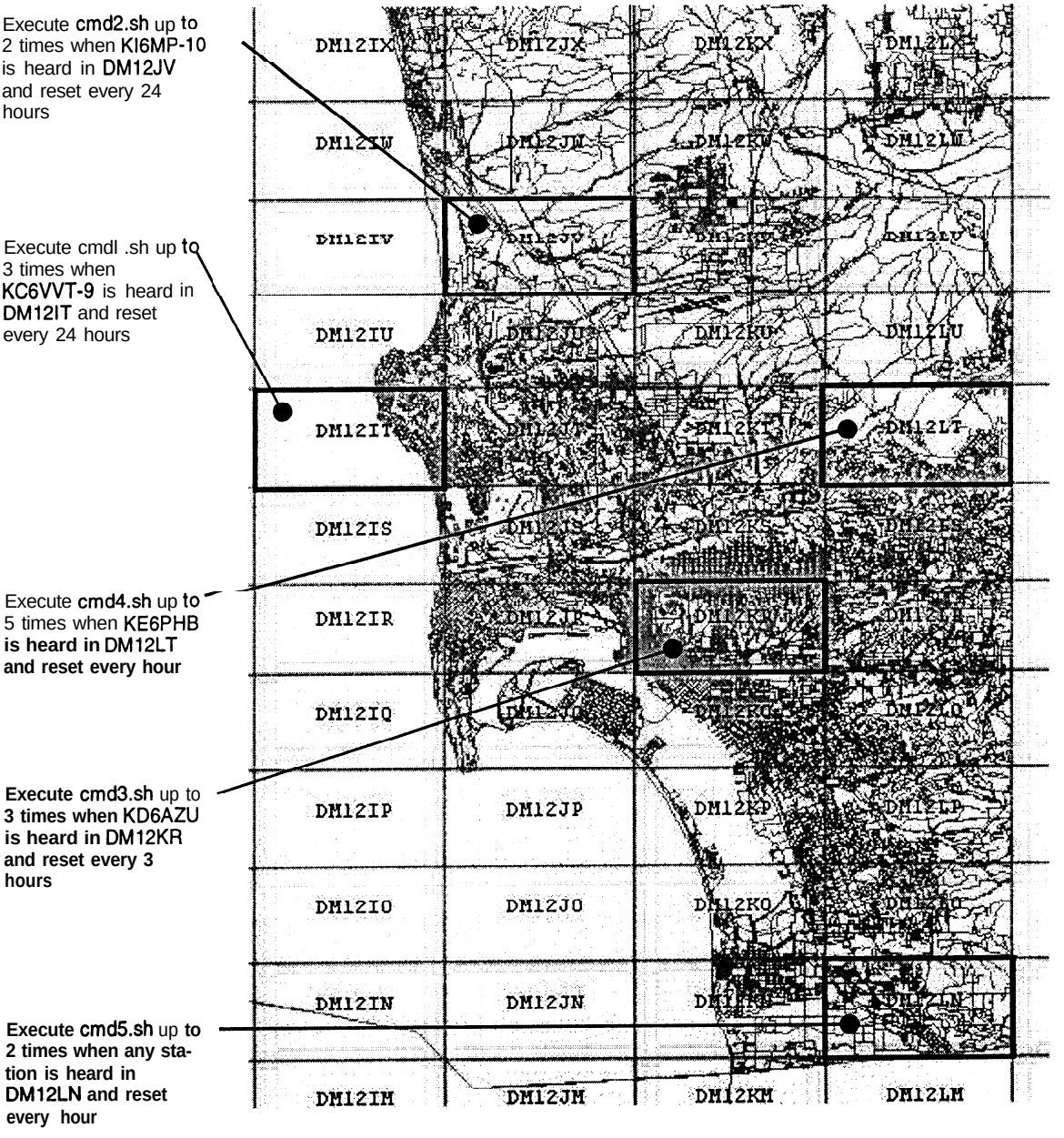


Figure 1 San Diego, California

PerlAPRS is a program written in the Perl computer language. Perl runs on all popular computer platforms today including MacOS, Windows3.1/95/NT, AMIGA, Unix, Linux, and many more. In addition, since Perl is compiled at run time, there is no need for a version precompiled or packaged for a specific platform. Also, since all source code is included for PerlAPRS, the user may easily alter the program to meet specific needs and is encouraged to do so. However, most users will find modifying the shell scripts rather than the program should meet most requirements.

PerlAPRS examines incoming packets from an APRS network and executes commands when a callsign and location match the criteria specified by the user. Location criteria is specified using grid squares. For example, when KC5PVL enters grid square DM12LW, a computer command specified by the user can be automatically executed.

Figure 1 shows grid squares overlaid on the city of San Diego, California, it will serve as the basis for the examples in this paper. The map shows several grid squares that are targeted for an action when the criteria specified in the *callsign.dat* file is met.

THE CALLSIGN FILE

When PerlAPRS starts, it reads the user's callsign database file with the default filename of *callsign.dat*. This file provides the list of callsigns that PerlAPRS is to search for. The file consists of one or more lines of text as shown in Figure 2. A separate line (record) is required for each callsign. Each line of the file is further broken into five fields, a separate field for each parameter.

```
KI6MP-10      cmd2.sh      DM12JV      2          1440
KC6VVT-9      cmd1.sh      DM12IT      3          1440
KD6AZU        cmd3.sh      DM12KR      3          180
KE6PHB        cmd4.sh      DM12LT      5          60
*             cmd5.sh      DM12LN      2          60
```

Figure 2 Example Callsign File

The first field indicates the callsign that PerlAPRS is to listen for. In the example, PerlAPRS will listen for KI6MP-10, along with KC6VVT-9, KD6AZU, and KE6PHB. The asterisk character, shown on the last line of the example, is a wildcard that means "any" callsign.

The second field indicates the command that will be executed when the callsign is heard. It can be any computer command, however, as we will discuss later, shell scripts are powerful and easily implement commands. In the example, *cmd2.sh* will be executed when KI6MP-10 is heard.

The third field represents the grid square in which the callsign must be heard. Returning to the example, PerlAPRS is listening for KI6MP-10 in grid square DM12JV.

The fourth field is provided to limit the number of times the command is executed during an active period. This parameter is necessitated by the repetitive nature of APRS packets. For example, if a station continues to broadcast packets while located within the grid square, the command would be executed each time a packet is heard. Since some APRS packets (e.g., mobile) are transmitted every few minutes, in many cases it would be undesirable to have the command executed repetitively in a short period. For this reason, the value is typically a small number (e.g., l-5). However, indicating a large value will cause the command to be executed virtually without limit. Conversely, if one wishes to disable execution, setting the value to zero essentially disables the command without removing it from the database. In the example, the command, *cmd2.sh*, will be executed no more than 2 times during an active period.

The last field is provided to allow the user to specify the active period. The active period is the time expressed in minutes in which PerlAPRS is actively listening for the specified callsign. This is important, since without a means of resetting the execution counter, it would be inconvenient to leave the program running for an extended period (e.g., many weeks).

The following scenario may help to illustrate the need for specifying the active period. Assume we wish to leave PerlAPRS running indefinitely. Also assume we don't want to execute a command every time a packet is heard since this could be hundreds of time during a 24 hour period. If we set the execution counter to a small value, we will limit the number of times the, command is executed. However, once that count is reached, commands will no longer be executed. The active period parameter is therefore provided to allow the user to specify when the execution counter specified in field 4 is to be reset. Returning to our example, we see that a command will be executed no more than 2 times in a 24 hour (1440 minutes) period for KI6MP-10. At the end of the 24 hour period, the counter is reset and the command can again be executed up to 2 times during the next active period.

When PerlAPRS begins it reads in the callsign file and shows the original information provided by the user along with the conversion of the grid square to latitude and longitude*. To completely describe the grid square, requires the latitude and longitude of the lower left and upper right points of the square. These comer points are used by PerlAPRS to determine if the station is within the grid square.

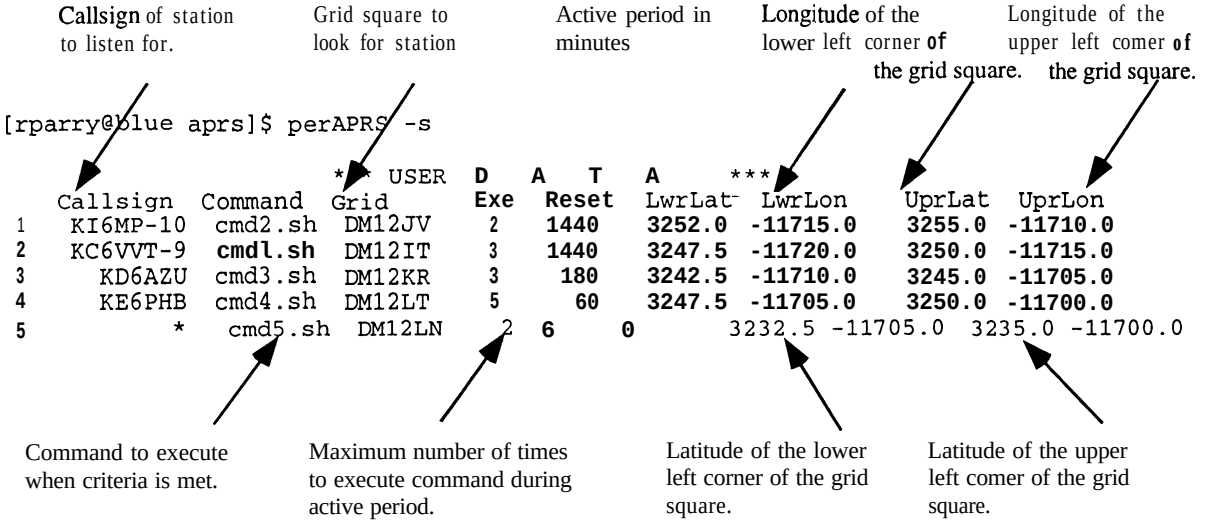


Figure 3 Output from PerlAPRS based on user's callsign file

SHELL SCRIPTS

When an APRS packet is heard, the command specified by the user is executed. It is important to emphasize that virtually any command can be executed, one is not limited to shell scripts. However, they are simple to write, flexible, and powerful. Scripts should meet the needs of most users. Alternatively, one can use Perl scripts which are even more powerful, yet still easy to write.

How to write shell scripts is a subject all by itself, and for this reason only a few examples are provided here. You don't have to be a software engineer to write scripts, but a knowledge of UNIX commands is important. The examples below were developed on a Linux system. These commands are not expected to work on other systems without customization. They are provided here as examples for illustrative purposes.

The following shell script will make a sound by sending the audio file *chirp.wav* to the audio output port of the system.

```
#!/bin/bash
cat /sounds/chirp.wav > /dev/audio
```

Shown below is a simple shell script to send email.

```
#!/bin/bash
echo "Match for KK5SU" | mail rparry@qualcomm.com -s "perlAPRS notification"
```

---

2 There is a common misunderstanding that for APRS applications latitude and longitude is specified as: degrees, minutes, and seconds when it should be: degrees, minutes, and decimal minutes. For example, 100 degrees, 40 minutes, and 30 seconds is written as 10040.500 and not 10040.30.

STARTING PerlAPRS

PerlAPRS is invoked from the command line like most UNIX style commands. Note that in the examples below, several command line arguments may be passed to the program which allow the user to alter PerlAPRS defaults.

`perlAPRS -h` `perlAPRS -help`	Display a ***help*** screen.
`perlAPRS -v` `perlAPRS -version`	Display the current version of the software.
`perlAPRS -s` `perlAPRS -show`	Normally PerlAPRS will not provide any output. The "show' option allows the user to see the progress of PerlAPRS. The *show* option will display only valid APRS packets that contain a position (latitude and longitude).
`perlAPRS -d` `perlAPRS -debug`	The ***debug*** option forces PerlAPRS to display packets that do not contain a valid position. This option is primarily for program debugging purposes.
`perlAPRS -d -s`	This example shows how PerlAPRS can be made to display both packets with and without latitude and longitude information.
`perlAPRS -p /dev/cua2` `perlAPRS -port /dev/cua2`	The default *port* that PerlAPRS will open is "/dev/cua1". However, you can specify an alternate serial port using this option. For this example, a Linux serial port name is indicated. For other platforms, consult the system's documentation.

`perlAPRS -p www.wa4dsy.radio.org:14579`
`perlAPRS -p sboyle.slip.netcom.com:14579`
`perlAPRS -p www.ne1h.radio.org:14579`
	If an Internet address is specified, PerlAPRS will open a socket to the address and obtain TNC data from the Internet. The examples listed are three presently know APRS servers. The text preceding the colon is the host name. The number following the colon is the port number which is fixed at 14579 for APRS applications

`perlAPRS -p trip.tnc`	A third specification for the *port* option is not actually a port, but a data file. If you have saved raw TNC packets to a text file, PerlAPRS will open the file rather than opening a serial port or an Internet communication's socket. This last variation of the port command is included for completeness, its main purpose is to allow PerlAPRS testing using known packets.
`perlAPRS -f callsign2.dat`	If a command *file* is not specified on the command line, PerlAPRS will use the default filename ***callsign.dat***. However, as shown in the example, the user can force an alternate command file to be called using the -f option.
`perlAPRS &`	This example shows how most users will run PerlAPRS. In this example, PerlAPRS will not show any output. It is also run as a background process by adding the ampersand character (*&*).

HOW IT WORKS

Figure 4 shows a sample output from PerlAPRS with the "-s" (show) option on. When a packet arrives, the callsign of the originating station is extracted along with the station's latitude and longitude. This information is then compared with each callsign in the database created by the user. If a match is found, the command is executed.

The first line in the example below shows a packet from originating station KD6AZU. PerlAPRS extracts the callsign, latitude, and longitude, and displays them on the line following the packet. Since this is a valid APRS posit packet, PerlAPRS will search the callsign database looking for a match for KD6AZU. As shown in the example, the first two attempts at a match fail. The third comparison is a match shown in bold print for illustrative purposes. The command, *cmd3.sh*, is then executed and the execution counter is incremented. Note that the first line of the match included the time that the packet was heard along with the maximum execution count specified by the user (e.g., 3). The second line of the match shows the time that the execution counter will be reset, the present value of the counter (e.g., 1) and the name of the execution command. When a second and third APRS packet arrives from KD6AZU, a match occurs and the command is executed again. When the fourth packet arrives, the command is not execution since the maximum execution count has been reached. No additional matches for KD6AZU will cause command execution. However, by the time the fifth packet arrives, the execution counter has been reset by the timeout and command is executed again.

```
Packet= KD6AZU>APRS,KD4DLT-7,N4NEQ-2,WIDE*:@042327/3243.70N/11707.70W/0
          KD6AZU       3243.700     11707.700
        - KI6MP-10     DM12JV
        - KC6VVT-9     DM12IT
        *   KD6AZU     DM12KR Sun Aug 10 15:56:13 1997    3
                                  Sun Aug 10 15:57:13 1997    1      cmd3. sh
          KE6PHB       DM12LT
                *      DM12LN
Packet= KD6AZU>APRS,KD4DLT-7,N4NEQ-2,WIDE*:@042327/3243.70N/11707.70W/0
          KD6AZU       3243.700     11707.700
        - KI6MP-10     DM12JV
        - KC6VVT-9     DM12IT
        *   KD6AZU     DM12KR Sun Aug 10 15:56:23 1997    3
                                  Sun Aug 10 15:57:13 1997    2      cmd3.sh
          KE6PHB       DM12LT
                *      DM12LN
Packet= KD6AZU>APRS,KD4DLT-7,N4NEQ-2,WIDE*:@042327/3243.70N/11707.70W/0
          KD6AZU       3243.700     11707.700
        - KI6MP-10     DM12JV
        - KC6VVT-9     DM12IT
        *   KD6AZU     DM12KR Sun Aug 10 15:56:34 1997    3
                                  Sun Aug 10 15:57:13 1997    3      cmd3.sh
          KE6PHB       DM12LT
                *      DM12LN
Packet= KD6AZU>APRS,KD4DLT-7,N4NEQ-2,WIDE*:@042327/3243.70N/11707.70W/0
          KD6AZU       3243.700     11707.700
        - KI6MP-10     DM12JV
        - KC6VVT-9     DM12IT
        *   KD6AZU     DM12KR Sun Aug 10 15:56:44 1997    3
                                  Sun Aug 10 15:57:13 1997    3      cmd3.sh
          KE6PHB       DM12LT
                *      DM12LN
Packet= KD6AZU>APRS,KD4DLT-7,N4NEQ-2,WIDE*:@042327/3243.70N/11707.70W/0
          KD6AZU       3243.700     11707.700
        - KI6MP-10     DM12JV
        - KC6VVT-9     DM12IT
        *   KD6AZU     DM12KR Sun Aug 10 15:57:14 1997    3
                                  Sun Aug 10 15:58:14 1997    1      cmd3.sh
          KE6PHB       DM12LT
                *      DM12LN
```

Figure 4 Output from PerlAPRS

MAIDENHEAD GRIDS

PerlAPRS relies on the use of grid squares to specify location. It accepts grid square parameters in 2, 4, or 6 letter formats. A 2 letter grid square covers such a large geographic area that the entire United States can be described in only 8 grid squares. A four letter grid square is smaller, but still represents a very large area (approximately the size of Connecticut). The 6 letter grid square is much smaller and is well suited for many applications in metropolitan areas.

It is not possible to describe the area of a grid square for a given format (e.g., 2, 4, or 6 letters) since they vary with their location on the earth. To illustrate the point, Figure 5 shows the size of a 2 and 4 letter grid square for the northern and southern portions of the United States. Even larger variations in grid square size occur between the poles and the equator.

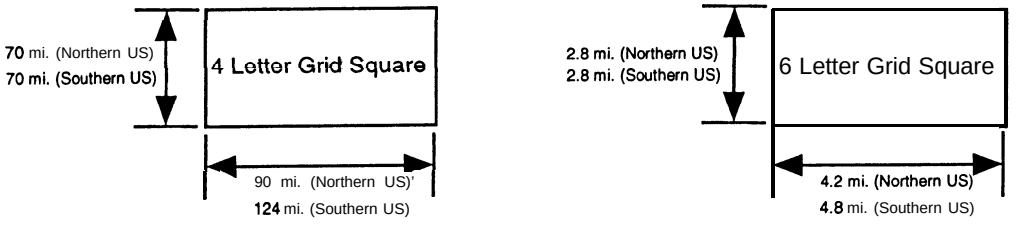


Figure 5 Grid Square Comparisons

Note that only the longitudinal distance varies between the southern and northern portion of the United States. The reason for this apparent anomaly stems from the fact that lines of latitude are parallel to each other and therefore separated by a constant distance. Lines of longitude are not parallel, they meet at the poles and are farthest apart at the equator.

CONCLUSION

PerlAPRS was developed to expand the usefulness of APRS to automated unmanned operations. It is an application that should prove useful in circumstances that require a specific action to a predefined packet specification. Using shell scripts or other languages developed by the user in conjunction with PerlAPRS should provide the framework for developing and extending APRS to many unique applications.

ACKNOWLEDGMENT

Thanks to Bob Bruninga, WA4APR; for permission to use the APRS trademark. Special thanks also to Keith Sproul, WU2Z; Mark Sproul, KB2ICI; and Steve Dimise, K4HG; for their pioneering work in this area.

---

[3] See ARRL web page listed in the bibliography for a more detailed explanation of this topic.

SYSTEM REQUIREMENTS

Linux was the development platform for PerlAPRS, however, it will work on any platform that supports perl version 5.002 or later. In addition, since Perl is available on virtually every popular computer platform, PerlAPRS should be able to be implemented easily. Any limitations are more likely to be with the platform's ability to support shell scripts. However, as emphasized in this paper, any computer language can be used to developing commands.

DISTRIBUTION

PerlAPRS is a freeware program available under the GNU General Public License, Version 2, June 1991, Free Software Foundation, Inc. 625 Massachusetts Avenue, Cambridge, MA 02139. It may be downloaded from the author's home pages at: **http://people.qualcomm.com/rparry/perlAPRS**. Further information regarding the necessary files to download and system requirements are included there.

BIBLIOGRAPHY

1. Bruninga, Bob, "Automatic Packet Reporting System (APRS)," 73, December 1996, pp. 10-19.
2. Dimse, Steve, "javAPRS: Implementation of the APRS Protocols in Java," *ARRL and TAPR 15th Digital Communications Conference Proceedings,* Seattle, Washington, September 1996, pp. 9-14.
3. Ford, Steve, "Where Am I," *QST,* April 1994, pp. 86-88.
4. Horzepa, Stan, "APRS Tracks: RELAY, WIDE, and Other Paths," *Packet Status Register,* Fall 1996 - Issue #64, pp. 30-3 1.
5. Horzepa, Stan, "APRS Tracks: Alias Envy," *Packet Status Register,* Summer 1996 - Issue #63, pp. 25-26.
6. Horzepa, Stan, "APRS Tracks," *Packet Status Register,* Spring 1996 - Issue #62, pp. 16-17.
7. Horzepa, Stan, "Getting On Track with APRS," American Radio Relay League,Newington, CT.
8. Parry, Richard, "Position Reporting with APRS," *QST,* June 1997, pp 60-63.
9. Wilson, Mark, "QST Compares: GPS-Compatible TNCs," *QST,* October 1995, pp. 68-71.

WEB SOURCES

1. The ARRL web page **http://www.arrl.org/locate/gridinfo.html** provides an excellent source of information on Maidenhead grid squares. The page also allows one to interactively determine a gird square from the latitude and longitude provided by the user.
2. To join the APRS mailing list, send email to **listserv@tapr.org** with **subscribe aprssig FirstName LastName** in the body of the message.

UPDATE ON DIGITAL VOICE TECHNOLOGIES

Paul L. Rinaldo, W4RI
4559 Quality Street
Fairfax, VA 22030
Email: prinaldo@arrl.org

Introduction

At the 1996 Digital Communications Conference, I presented a paper on "Amateur Radio Digital Voice Communications" with the intent of promoting interest among amateur experimenters. Not much progress has been made in developing amateur digital voice systems during the past year. Industry is still doing developmental work but standards are not easily achieved.

Spectrum Efficient Digital Land Mobile Systems for Dispatch Traffic

The 1996 paper appended an ITU-R draft new recommendation with the above title. It provided core parameters for VHF/UHF digital voice systems known as Project 25, TETRA, IDRA and DIMRS. At a Geneva meeting of ITU-R Study Group 8 (mobile, radiodetermination, amateur and related satellite services), this draft recommendation was delayed over a debate concerning intellectual property rights, and whether to include a French system TETRAPOL and an Ericsson technique known as EDACS. There were also objections expressed over the desirability of publishing recommendations having multiple annexes, the ideal being one global standard. (There are fundamental differences in views between Europe and the U.S. over single vs multiple standards, the U.S. view being "let many flowers bloom" or "let the market place decide," the European position being that it makes sense to have one world standard--preferably theirs.) These issues are to be brought before the ITU 1997 Radio Assembly in October.

Maritime Mobile Digital Voice Systems

ITU-R Study Group 8 took the view that the maritime mobile community should not develop its own digital voice standards but wait for a common land mobile standard to emerge. While the land mobile services could tolerate different digital voice standards in different parts of the world, this is not the case for ships which should use only one global standard for VHF/UHF.

HF Digital Broadcasting

HF broadcasting has used double sideband amplitude modulation (DSB AM) since its inception. Receivers are abundant and some are quite inexpensive. All is not well because HF radio propagation introduces multipath fading distortion. Furthermore, the other radio services using the HF bands have been urging the HF broadcasters to switch to single sideband (SSB) with reduced carrier to conserve bandwidth. Proper reception of RC-SSB requires synchonous detection, which tends to make receivers more expensive, at least until the day when critical mass has been reached. It's been a chicken-and-egg situation: HF broadcasters won't change to SSB until receivers abound; manufacturers can't reach the required economy of scale until all the broadcasters change to SSB.

HF SSB broadcasting may be at a dead end and broadcasters are now thinking of leap-frogging this technology and going digital. The problem is that the standards have yet to be developed and research still remains to be completed.

Fortunately, there is a development effort underway. Significant research work is being done by the NASA Jet Propulsion Laboratory under contract with the Voice of America. A wide number of organizations are interested: Motorola, National Association of Shortwave Broadcasters-USA, Telediffusion de France, RAI, Harris Corporation, Radio Nederland, Continental Electronics Corp, Telefunken, BBC, Sony, Australian Broadcasting Corporation, BBC, Detche Welle, and Sangean Corporation, to name some of the major players.

The technical challenge is difficult enough. The economics of HP digital broadcasting remain to be solved. The question remains: Why is it thought possible to get people to buy digital radios when they can't afford synchronous SSB receivers?

During 1996, VOA ran some on-the-air digital broadcasting tests from its Delano, California facility to Washington, DC. While there were problems, the results were sufficiently positive that further design was justified. The audio encoder chosen for these tests was the AT&T G728 (LD-CELP), which had been optimized for speech but not for music. Various modulation schemes up to 8PSK were tried and there are plans to experiment with modulation levels higher than 8PSK during the latter half of 1997.

The results of the VOA/JPL task may or may not be applicable to amateur HF digital voice. The general objective is the same--transmitting a digital voice signal and receiving it with results that are better than for AM DSB. However, some of the parameters are different, notably that an amateur digital voice signal would probably have to occupy no more bandwidth than an SSB signal, while HF digital broadcasting could use bandwidths comparable to those for DSB AM. Broadcasters have a lot of high power Class C finals; amateurs these days have linear power amplifiers designed for SSB.

We don't need to transmit music; they do. Broadcasters must design to low-cost receivers
intended for mass markets; amateurs can accept a design that costs a little more.

Amateur HF Digital Voice Design Objectives

- Like for SSB, it would desirable for an amateur HF digital voice radio also to be usable for commercial/government use, for reasons of economy of scale.
- Transmitter power should be comparable to that normally used for SSB, i.e., 100 watts. External linear amplifiers would be used when additional power is required.
- The recovered audio should be noticeably better than for SSB over a fading path.
- Some type of signal processing should be available to favor the desired signal and reject an interfering signal, whether in a two-way contact or in a net.
- The bandwidth should not exceed 3 kHz and preferably not more than that of existing amateur SSB, or about 2.4 kHz.
- In addition to digital modulation and demodulation, a. normal SSB mode should be provided.

Conclusion

HF digital voice experimentation is an area where amateurs can make a significant contribution to the radio art. HF is sufficiently different from VHF/UHF because of a channel subject to multipath fading and interference. The Amateur Radio requirement is somewhat different than that of HF broadcasting to at least modify the technical parameters and possibly use different design strategies.

Using a PC and a Soundcard for Popular Amateur Digital Modes

Thomas M. Sailer, HB9JNX/AE4WA
Weinbergstrasse 76
CH-8408 Winterthur
Switzerland

Abstract

Recently, standard personal computers (PCs) have become powerful enough to do serious digital signal processing (DSP) without the need for a specialized DSP coprocessor. A standard PC soundcard serves as the interface between the analog world of the radio and the digital world of the PC processor. This equipment together with an appropriate software package allows the ham to operate many popular digital modes without a TNC.

Historical Perspective

Processing digital signals with a general purpose digital computer is not new at all. But only recently personal computers (PCs) have become fast enough to do serious realtime digital signal processing. A highend 486 or Pentium class machine is required for these applications. Analog interfaces (soundcards) with good quality are also relatively new; early models suffered from low resolution, low sampling rates and high prices.

Amateur modes having no timing relationship between receiving and transmitting, such as RTTY, FAX, SSTV or POCSAG, can be implemented quite easily. Standard operating system drivers for the soundcard may be used, and neither huge buffering nor long latencies do hurt. Several programs handling these modes popped up recently on the internet.

Modes requiring fast switching between reception and transmission, low latencies and an exact timing, such as packet radio and the synchronous shortwave data modes like Amtor and Pactor, are more difficult to implement. This paper reviews the problems associated with these modes and presents possible solutions.

After experimenting with digital signal processors, I started implementing a packet radio modem for PC's running DOS in late 1994. The development platform at that time was a 486DX2/66. A first version was presented at the Darmstadt, Germany, meeting in 1995 [Sai95]. The DOS version was continually developed further, got modularized, that is, the soundcard drivers and modem code was separated. The code was ported to Linux in early 1996 [Sai96], and back to Windows95 in late 1996 [Sai97].

The soundcard packet radio software is now available free of charge for amateur radio applications. The DOS and Windows95 versions are available from the PC/FlexNet homepage (see below), while the Linux version is included in the standard version 2.1.x kernel sources. Version 2.0.x users need to get the patch from the zone FTP site (see below).

Motivation

The software modem provides several advantages over the conventional approach using a TNC.

- Since there is no dedicated TNC hardware, this solution is cheap. Terminals are not widely used anymore, a PC is common in typical HAM shacks, and you will need it anyway to run those fancy graphical terminal programs. Todays PCs are shipped with soundcards already installed, furthermore soundcards go for as little as $30. Therefore, almost all the required hardware is already available at the typical ham shack.
- Easy diagnostics. No more fiddling with oscilloscopes, just run the diagnostic application to plot the input signal, eye diagrams etc.
- Software configurable. It is easy to change the operating mode without a hardware change with only a few keystrokes. With a conventional TNC, you have to buy and install another modem.
- Compared to specialized DSP processors, PC development tools are usually much more mature; the modem code may be single-stepped, data may be logged to disk, or even multiple modems connected by a software radio channel simulator may run simultaneously on a single machine, which greatly simplifies debugging and offers new development possibilities.
- Mobile operations: a contemporary laptop computer with builtin sound hardware and a handheld transceiver is all you need to operate on the road.

Of course, the software solution also has a few disadvantages.

- It requires host CPU processing power The requirements are quite moderate, however" A typical host CPU load is 10% of a Pentium75 (1200 Baud AFSK). The value scales nicely with CPU clock frequency. MMX is neither required nor used, since its moderate benefits hardly justify the pain to develop MMX aware applications.
- The modem software ties up the soundcard. It is however possible to operate two soundcards in a single PC.
- Audio quality of the soundcards is widely varying. The general trend seems to be the audio quality being inversely proportional to price. Cheap cards with just a single chip from Analog Devices or Crystal Semiconductors usually do fine, while the rather popular models from Creative Labs offer less quality. Their mixer chip's bass and treble controls introduce phase distortion. Because of the widely varying signal impedances, levels and connectors it is impossible to draw a generic wiring diagram to the radio.
- It is still not completely free of dedicated hardware. Soundcards usually do not have DC coupled outputs. Therefore, a simple interface circuit is required to connect the radio's PTT line to another PC port. The simplest circuit consists of a resistor and a transistor, plus the required connectors. Circuit diagrams of example circuits may be found on the FlexNet homepage.

Packet Radio

Packet radio requires a fairly fast switching time between receive and transmit, in the order of milliseconds. If this requirement is not met, the channel access algorithm, namely CSMA or eventually DAMA, rapidly shows performance degradation, leading to excessive collisions on the channel [Wel97].

Unfortunately, neither standard Windows95 sound drivers nor the Linux sound driver offers this performance. Actually, the Linux driver is „almost there“. An experimenal user-mode implementation of the packet radio modem using the Linux sound driver performs well on some computers, but suboptimum on others, depending on the CPU speed and network activity, etc. DOS does not provide a sound driver at all anyway.

Therefore, the packet radio software has to provide its own soundcard driver. This limits the supported soundcards to the vast majority of the Soundblaster or WSS' compatible cards. Since the driver requires direct access to the hardware, it has to be implemented as a kernel module under Linux, or as the Windows equivalent, a VXD.

Both the standard operating system sound driver and the packet radio modem driver want exclusive access to the hardware. This complicates installation. Details on the installation procedure can be found on the FlexNet homepage for the DOS/Windows95 case or in the AX25-HOWTO in the Linux case.

The DOS/Windows95 driver uses PC/FlexNet [Jos95a,Jos95b] as its AX.25 stack. This means that every application supported by FlexNet or its compatibility modules can be used. This is the vast majority of the existing packet radio programs. There is no need for special terminal programs.

Under Linux, an AX.25 stack is incorporated into the kernel networking. Therefore, the natural choice was to implement the soundcard modem driver as a standard Linux networking interface. While slightly more complex to install, other AX.25 stacks such as *NOS can be used as well.

Popular HF Protocols

Popular HF protocols, such as AMTOR (SITOR) [CCI86] or Pactor 1 [HS90], pose a few additional problems that have to be addressed.

HF protocols have very stringent timing requirements. CCIR recommends less than 20ppm clock deviation for SITOR! There are a few different possibilities to derive a clock signal in the PC:

1. the soundcard sample clock
2. the CPU clock feeding the cycle counter
3. the system timer
4. peripherial clock generator, such as the baud rate generator of a serial interface

[1] Windows Sound System; a hardware standard initially designed by Microsoft; it has nothing to do with the -availability of Windows drivers

While 1 is the natural choice for full duplex soundcards, it cannot be used on the half duplex variety, due to the discontinuity during switching between receive and transmit. 2 is a convenient source because it can be read easily and provides fine granularity, but is unfortunately only available on Pentium class machines. 3 might be inconvenient because of the games the operating system plays with the system timer. 4 ties up additional hardware resources otherwise not needed.

Neither of these sources fulfill the 20ppm requirement. The problem is not the stability, at least not if the PC is operated indoors, but the initial frequency. After all, a 100MHz P5 system may as well run with 100.1MHz or 99.9MHz. Therefore, a method to calibrate these clock sources is required. Now the only method to measure the frequency of a signal is to compare it to a signal with known frequency. Such a reference signal has to fit into the passband of the soundcard. I have experimented with the following signals:

- Longwave timecode transmitters. DCF77 can be received easily throughout central europe at 77.5kHz, and its pseudo noise phase code allows accurate measurements within a few minutes.
- TV broadcast horizontal line sync. VCRs with baseband video output are readily available and some TV stations have horizontal sync frequency derived from an atomic clock, such as the second german state network (ZDF).

GPS could also be used, but its outputs, the second clock and eventually the 10MHz reference output, do not fit into the passband of a soundcard, therefore some additional circuitry would be required.

Additionally, the „search space“ where to look for signals is much bigger than in the typical packet radio case. While in a typical packet radio mode the only unknown is the starting time of a transmitted packet, in an HF environment a station may be calling in any one of several major modes (such as Amtor or Pactor), each of which may consist of several variations (such as inverted or not in Pactor), and with an unknown offset from the receiving station’s carrier frequency.

This requires that the standby station uses many demodulators in parallel to dig for possible signals, which leads to the somewhat paradoxical situation that the standby operating mode requires much more CPU power than an ongoing circuit. The user may however limit the search space in exchange for CPU load, such as by limiting the maximum allowable frequency offset. The development of such a HF engine running under the Linux operating system is in progress. Most needed now is a decent terminal program; anyone wanting to volunteer is asked to get in touch with the author.

Outlook

I am planning to continue developing the available software by adding new modes and adapting it to new hardware when it gets widespread, such as AC97.

I am also planning to bring the benefits of the software modem approach to packet radio nodes. The target platform is likely to be the next generation RMNC/FlexNet hardware.

Web Resources

Analog Devices, Inc. http://www.analog.com

Crystal Semiconductors	http://www.crystal.com
Creative Labs	http://www.creaf.com
PC/FlexNet	http://home.pages.de/~flexnet
Linux AX.25 utilities	ftp://zone.pspt.fi/pub/linux
Author's Ham page	http://www.ife.ee.ethz.ch/~sailer/ham/ham.html

Bibliography

[CCI86] CCIR recommendation 476-4, Direct Printing Telegraph Equipment in the Maritime Mobile Service, 1970- 1986

[HS90] H.-P. Helfert, U. Strate, PACTOR - Einführende Protokollbeschreibung (Level 1), 5. November 1990

[Jos95a] Gunter Jost, An Introduction to FlexNet, 14th ARRL digital communications conference, Arlington, TX, 1995

[Jos95b] Gunter Jost, PC/FlexNet - Die neue Plattform fur PR-Anwendungen, 11. Internationale Packet-Radio-Tagung, Darmstadt, 1995

[Sai95] Thomas Sailer, FlexNet-Workshop: Die DSP-Treiber, 11. Internationale Packet-Radio-Tagung, Darmstadt, 1995

[Sai96] Thomas Sailer, „cheaper packet" mit Linux, 12. Internationale Packet-Radio-Tagung, Darmstadt, 1996

[Sai97] Thomas Sailer, „PacketBlaster 97" - Soundkarten-PR mit aktuellen Betriebssystemen, 13. Internationale Packet-Radio-Tagung, Darmstadt, 1997

[Tuc84] Tucson Amateur Packet Radio Corp., AX.25 Amateur Packet-Radio Link-Layer Protocol Version 2, TAPR, October 1984

[We197] Matthias Welwarsky, Kanalzugriffsverfahren im Packet Radio, 13. Packet-Radio-Tagung, Darmstadt, 1997

Terminal Node Controllers

Towards The Next Generation ?

A possible basic architecture for future TNC designs.

Darryl Smith, BE, VK2TDS
POBox 169
Ingleburn NSW 2565
Australia
VK2TDS@OzEmail.Com.AU
+61298295414(H)
+61 2 9268 7890 (Fax)
+61 4 1292 9634 (Mobile)

ABSTRACT

This paper describes work into a new generation of hardware for Terminal Node Controllers (TNC's). This development has been done under Linux on IBM compatible hardware, but is easily transferable to a more traditional microprocessor based TNC design.

INTRODUCTION

Have a look about my shack and you will find an amazing number of TNC's. There is an original TNC 1, a couple of home built TNC-2's in various states of repair, another TNC-2 board with no parts and a KPC-9612 dual port TNC.

With the exception of the KPC-9612, they are all single port designs. The KPC-9612 is a dual port design, but as far as I can tell it implements the serial ports using software rather than hardware limiting further expansion.

These TNC's are also limited in their bit rates. I doubt that any of the designs could exceed 19.2 kBit/sec.

Higher speed equipment is needed if amateur radio is to survive. This paper describes my work into assisting those designing high speed radios and modems.

'What works is what matters' and through my use of a development board, I have proved that this solution will work.

My work will not suit those working on really high speed links but is ideal for applications up to 100 kBps.

TNC-TNG MAILING LIST

Much of the initial work on this project was done as a result of discussions on the TNC-TNG mailing list at LANTZ.COM.

All the participants have gone out of their way to help in this project, exemplifying the spirit of Ham Radio. Some of the suggestions made are included here whist others have been discounted for various reasons.

Early on I made the decision that more intelligent hardware was required if we were to proceed. Using IC's designed 2 years ago would not help with 10 year old TNC designs.

MOTOROLLA 68360

The 68360 communications controller from Motorolla is one IC that initially seemed most suitable for a new TNC design. With a 32 bit CPU, 4 sync/async ports, 2 more async ports, and a programming port it is quite suitable for use within a TNC. With the 68360EN, one of the sync/async ports becomes a full Ethernet port, with very little internal hardware required.

The 68360 contains almost 2Kbytes of memory for pointers to packet data structures necessitating little host intervention during packet reception.

However, several things led me away from using this IC. Firstly, development boards were too expensive, with all boards I could find costing over US$2,000. In addition, low quantities of this chip were approaching the US$lOO mark, without guaranteed supply.

CIRRUS CD-2341

Once these problems became obvious, one participant of **TNC-TNG** suggested the use of the Cirrus Logic CD-2400 series of chips. When I received the data books from Cirrus Logic I was sold.

Cirrus call this IC family 'Advanced Multi-Protocol Communications Controllers', and they are truly advanced. The chip has the following capabilities

- Four full duplex ports each running up to 134kBits/second, with two independent bit rate generators per port.
- Support for async, async-HDLC along with synchronous HDLC & SDLC on all channels
- 32 bit address, 16 bit data double buffered DMA controller for each transmitter and receiver.
- NRZ, NRZI and Manchester coding.
- a Digital PLL on each receiver and two timers per channel
- PPP, SLIP and MNP-4
- Intel and Motorolla hardware and software compatibility.
- 16 byte receive and transmit FIFO's

To top off that list they are also quite cheap at less than US$25.00 in small quantities.

Hardware

If this chip were used along with a microprocessor, designing and manufacturing a TNC with 4 or 8 radio ports, each capable of high speed operation, would not be a complex task. A TNC could literally consist of a microcontroller, memory, CD-2431 and maybe an Ethernet chip.

Development hardware is however scarce for this chip. After searching for months I finally found a suitable board. It is hoped that this paper spurns some activity in designing a development board even more suited to amateur radio use.

THE CRONYX-SIGMA 22 DEVELOPMENT BOARD

The development board I eventually chose was the Cronyx Sigma-22 at only US$340 plus postage. This is not cheap, but I feel that the board is well worth it.

Although documentation for the card is lacking the technical support that I received was remarkable. Unfortunately, although all the normal manuals were supplied with the card, they were all in Russian. I understand that Cronyx is working on an English translation.

Drivers are available from Cronyx for Windows, Linux and BSD/386 Unix. The software supplied does is not suitable for amateur use. For this reason, I have rewritten much of the Linux driver. The changes I have made are centred in adding support for timer delays and passing full frames to other networking layers without processing.

*

The Sigma-22 has two external serial ports at RS-232 level and two more on the internal connector at TTL levels. Appendix A documents the external pinouts for each port, whilst Appendix B shows the internal pinout of the otherwise undocumented internal connector. Like the rest of the board thought has been put into the pinouts of this connector. Should you connect the plug around the wrong way, the only consequence will be that ports 2 and 3 will be swapped. (Ports 0 and 1 are the external ports).

This internal connector is designed and positioned so that it is possible to have a daughter board inside the computer with a couple of 1200 bps modems with PTT hardware, leaving RS-232 level outputs on the other ports for higher speed hardware.

When my Sigma board, the 40 pin header was missing. It was quite simple to insert a 40 pin header from my junk box.

All the signals are at TTL level so interfacing is simple. If connecting to the external connectors is required at TTL levels, level converters are required.

HARDWARE

Essential Pins

Today there are three main physical interfaces used to connect to equipment such as modems. They are TTL, RS-232 and RS-422/RS-485. Although I prefer working with TTL I can see that some users will prefer one of the other signal levels.

A watchdog timer for PTT lines is essential in the TNC, as is a way to disable it, without opening the back of the controller. In a previous design I implemented this feature by allowing two pins in the connector to be shorted disabling the watchdog.

I feel that RTS/CTS, as well as DCD and data should be available with RS-232 levels, allowing 1200 bps voice modems to be connected without any glue logic.

Clock Signals

After dealing with the normal array of SCC chips, the CL-CD853 1 is a nice change. Gone are the days when bit rate generators were scarce and much anguish took place over the hardware configuration of clocks.

Clock inputs and outputs are available for each transmitter and receiver on the chip, with programmable divisors. Thus modems with a 32* receive clock interface with the same ease as a modem requiring a 1* or 16* clock.

Unlike 8530 designs no external divider is required to loop the transmit clock into the receiver, as the divider is available inside the chip.

Transmit Clocks

The only glaring problem I could find with the CL-CD243 1 is the lack of a transmit clock at a rate other than unity. This limits the use of modems such as the K9NG 9600 bps modem which require a clock sixteen times the bit rate (Unless you have purchased a modem such as the TAPR 9600 bps unit and 'Clock Option').

If a full design were produced using this chip it may be desirable to use add an LSI bit rate generator. However in most cases the clock recovery in the IC will be good enough so that external DPLL's ,Which is the main use for these clocks anyway, are not required.

The receiver can accept input clocks at any integer multiple of the input clock. Appendix C contains an evaluation of where present modem designs would need to be modified to work with this chip.

SOFTWARE

To check the viability of using a Cirrus communications controller in a production design I have prototyped the software under Linux. The software is very basic and implements a simple TNC. At the time of writing, not even the timers were implemented, but transmitting and receiving data works flawlessly.

Why KISS?

The best way to learn what I did in software is to use the source - both the original and my modified version. Unless you are familiar with the Linux SKB buffer system I suggest you look at Alan Cox's excellent article[l] in the Linux Journal.

The Cronyx Sigma Linux drivers are based on the same underlying structure as Linux in that it uses sk_buf's rather than mbuf's common to BSD and NOS programmers. Sk_buf 's are based on the presumption that memory is cheap, and the overhead of buffer assignment inside an Interrupt Service Routine is expensive. Therefore s k_bu f's are assigned to a size slightly larger than the average packet size, and. can be chained

Th driver however assumes that each s k_bu f contains only one packet meaning that if the pre-assigned buffer is smaller than the incoming packet then that packet is lost.

This is not a problem with the transmitter as an s k buf 's can be allocated large enough. In practice since the AX-25 protocol required to be used in many countries the maximum packet size is known.

Transmit Delays

In packet radio there are several delays required for correct operation. They are TxDelay, TxTail and SlotTime. Luckily none of these delays overlap allowing a single programmable timer to be used on each channel.

I will be implementing these delays using one of the general timers on each channel in the CD-243 1. According to the documentation, general timers on the CD2341 can be set under 1 mSec and still

maintain accuracy. For simplicity I will be setting the timer base unit to one mSec in software, with all timers being a multiple of this.

Using a 1 mSec granularity allows for a timer of over a minute without loss in accuracy or additional hardware or software, which is fast enough for even the slowest HF radios. [1]

Testing

Whilst testing I like to remove as many variables as possible, as well as cause as little impact to those around me[2]. The first variable to remove was the radio interface, including modems and radios.

The KPC-9612 has an unusual feature on the 9600 bps radio port; it has digital input and output signals. These could be directly connected to the TTL signals on the Sigma board. OK, this is not entirely correct! The signals on the KPC-9612 are in fact scrambled[3] like all digital signals should be going to a data radio. However none of the documentation in the KPC manual mentioned this.

I was going to say that the KPC-9612 enabled me to ensure that everything was working before touching a radio. However it did not turn out that way. In the end I needed to start up my TNC-1 as a source of packets.[4]

The only thing that I could find that was lacking in the TNC-1 was that I was unable to ignore CRC errors. This is essential when trying to debug new hardware and software. I eventually saved the incoming kiss stream received from the TNC through the Cirrus chip, and replayed this back to the TNC to get the transmit running for the first time.

I noted something else whilst attempting to make sense of the data coming through the cirrus chip from the KPC. I had for some reason believed that the call signs in an AX.25 frame would be in ASCII. They are not.

This will not be a surprise to those who have read and understand the protocol definition. All call signs in AX.25 are standard ASCII upper case characters, BUT they are then bit shifted up by one bit.[5]

CONCLUSION

Through my work with the Cronyx board and the Cirrus CL-CD2431, I have proved that this chip is suitable for use in amateur radio. I have already demonstrated that writing software to support this IC is much easier than many other chips. I have also shown that CPU requirements are reduced considerably with the CD2431's in built DMA controller. However potential users should note that clock outputs are limited to the set bit rate of the channel

[1] I Hope ☺

[2] Whilst I was completing my University Thesis I noticed that a friend, who was also working on his thesis, was transmitting one packet every 1-2 seconds for a few hours. Suspecting a bug, I rang my friend and had to leave a message on his answering machine. He got back to me the next week, when he returned from a weekend away. In the meantime he had transmitted too many packets to count as his software was not quite as bug free as he thought.

[3] See 'Wireless Digital Communications' from TAPR for an explanation of scrambling.

[4] I should thank the designers of the TNC-1 for their forethought in adding an internal modem connector to the TNC-1. It was a work of genius, even if there are one or two features missing.

[5] I know this is not a revelation to many, but many Amateurs such as myself will fall into this trap.

In the past, most serial controllers used for amateur radio have been based on the ubiquitous 8530. This was understandable with so much software available for it.

I hope that through this project and paper that I have opened new doors to those developing for Amateur Radio.

Source Code

The source code as well as full documentation is available at `HTTP://www.ozemail.com.au/~vk2tds` Some of the software is available from Cronyx at their FTP site.

The software that is supplied with the Cronyx Sigma-22 card implements Frame Relay, Cisco and PPP packets.

Future Software and Hardware

This project has proved that a design based on the Cirrus CD-243 1 is quite viable with following areas of investigation suggested.. .

- Design of a microprocessor TNC - ideally including an Ethernet port.
- Transfer the code from Linux/Unix to a microprocessor architecture. Actually rewrite all the code.
- Use PPP rather than KISS for connections between the PC and the TNC.
- Add the complete AX.25 and maybe TCP/IP stack to the drivers.
- Maybe even implement a WWW server for configuration!
- Generate a 16 times transmit clock. This could then be implemented in a PAL or GAL.

Acknowledgments

I would like to thank all those who assisted me with this project. In particular I need to thank the following.

- Terry Behan, VK2TLU
- Craig Small, VK2XLZ (csmall@gonzo.triode.net.au)
- Cirrus Logic
- Serge Vakulenko of Cronyx. (Vak@cronyx.ru)
- TAPR and the designers of the TNC-1 and TNC-2.
- David Kelly, N4HHE, (dkelly@hiwaay.net)
- All the participants on the TNC-TNG Mailing list.

In addition I would like to thank all my friends at Pacific Power for their assistance.

REFERENCES

1. Kernel Komer: Network Buffers And Memory Management by Alan Cox, Issue 29, The Linux Journal, September 1996.[6]

2. The KISS TNC by Mike Chepponis, K3MC and Phil Kam, KA9Q

3. Multi-Drop KISS operation by Karl Medcalf, WK5M, 10th CNC, p 109-l 11

4. Linux Kernel Hacker's Guide. Now available at http://www.redhat.com

5. Computer Networking Conference Proceedings, 1984-l 996, ARRL & TAPR.

6. Cirrus Logic Data Book Cl-CD2431, Version 3.0, August 1996, Cirrus Logic.[7]

[6] This article can be found on the Internet at http://redhat.com.au

[7] NOTE: If you are writing software for the CD-243 l you should also specifically ask for the errata sheets from Cirrus Logic.

7. HAPN-2 by John Vanden Berg, VE3DW, CNC-11, Pp.98-105

8. PI Card, Dave Perry, VE3IFB, CNC-10, Pp.121-124

9. A prototype TNC-3 Design Approach, CNC-12, Pp.1 1-15

10. A proposal for a standard digital interface, Jeffrey Austen, A9 JA, CNC- 13, Pp. l-4

Appendix A

HDB-26	V.35	RS-232
1	T_xDa	
2	T_xDb	
3	R_xDa	
4	GND	GND
5	$R_xC_{in}a$	
6	select	select
7	$T_xC_{out}a$	
8	GND	GND
9	$T_xC_{out}b$	
10	RTS	RTS
11	GND	GND
12		T_xD
13	GND	GND
14		R_xD
15	$R_xC_{in}b$	
16	CD	CD
17		T_xC_{in}
18	GND	GND
19	DTR	DTR
20		T_xC_{out}
21	CTS	CTS
22	$T_xC_{in}a$	
23	$T_xC_{in}b$	
24	R_xDb	
25	DSR	DSR
26		R_xC_{in}

Appendix B

GND	1 2	RTS,
+12 V	3 4	R_xD_2
+5 V	56	$T_xC_{lk}In_2$
GND	7 8	$R_xC_{lk}In_2$
-12 V	9 10	T_XD_2
GND	11 12	
	1 3 1 4	$T_xC_{lk}Out_2$
CD_3	15 16	
DTR_3	17 18	CTS,
DSR_3	19 20	
	21 22	DSR_2
CTS_3	23 24	DTR_2
	25 26	CD_2
$T_xC_{lk}Out_3$	27 28	
	29 30	GND
T_xD_3	31 32	-12V
$R_xC_{lk}In_3$	33 34	GND
$T_xC_{lk}In_3$	35 36	+5 V
R_xD_3	37 38	+12 V
RTS_3	39 40	GND

Pin outs of the Cronyx internal feature connector.

Appendix C

KD2BD PacSat Modem (1200)	Requires 16* clock although this is divided to a 1* clock anyway.
TAPR/K9NG 9600	16 or 32 times. Used for clock recovery. It should be possible use the IC for clock recovery.
HAPN-4800 and derivatives	No clock signals required.
TCM-3 105	No clock signals required
7 109 World Modem Chip	No clock signals required

CONTACTS

CRONYX LTD (Russia)

Tel. (7-095)939-23-23
EMAIL: info@cronyx.ru
WWW: http://www.conyx.ru

Many people will however find it easiest talking to their USA agents.

TETRA GROUP

USA Agent for Cronyx

Joseph Kulinets
(Joseph_Kulinets@compuserve.com)
Tel: (203) 968-9158
fax: (203) 968-94 18

Cirrus Logic(USA)

3 100 West Warren Ave
Freemont, CA, 94538
Tel: (510) 623 8300
Fax: (510) 252 6020

On-air Measurements of CLOVER P38 Throughput

Ken Wickwire (KB1JY), Mike Bernock (KB1PZ) and Bob Levreault (W1IMM)

1. Introduction

This paper is one of a series treating on-air measurement of throughput of various HF data-transmission protocols available to amateurs (see the references to our other reports at the end of the paper). Here we describe an extensive set of measurements of throughput for text and other files sent over near-vertical-incidence-skywave (NVIS) and one-hop skywave (OHS) paths using the file transfer protocols implemented in the HAL P38-CLOVER terminal package. (NVIS paths, which often produce difficult channel conditions, are used to communicate over 20- to 300-mile ground distances using antennas that can launch energy at high takeoff angles.) The measured throughput data in our experiments were analyzed using software specially written to compute throughput statistics from our CLOVER data.

The HAL CLOVER modem and data-transmission protocols have now been in use for about five years and the system has established itself as one of the most reliable, inexpensive and efficient means of HF data transmission available to hams. Several versions of CLOVER are available (the PCI4000, DSP4100, P38, and others). The versions are distinguished mainly by their maximum speed of data transmission (and corresponding modulation complexity), which is a function of the processing power of the associated hardware (computer or modem or both).

The P38 modem is a low-cost (less than $400) PC-card version of CLOVER that has a slower TI microprocessor than the Motorola processor used by its more expensive (around $1000) relatives, the PC14000 and DSP4100. The P38 does not have the computing power to process received signals in the 8P2A (8-phase, 2-amplitude) and 16P4A (16-phase, 4-amplitude) signaling modes that the other versions can use. Otherwise, the P38 can do everything the PC14000 and DSP4 100 can do, including uncompressed and compressed file transfers using the CLOVER automatic repeat request (ARQ) protocol. Using the compressed mode, one can send graphics and other “binary” (eight-bit-character) file data, in addition to text files.

CLOVER changes its modulation mode (varying the number of bits per symbol while keeping the symbol *rate* fixed at 3 1.25 symbols per second) in response to changing channel conditions. Channel conditions are assessed (at receivers) by measuring the amount of error-correcting capacity [ECC] used, the number of erroneous data blocks and the phase dispersion [PHS], among other things. Receivers send this information to transmitters at turnarounds, and in Version 23 of the firmware, the transmitter makes decisions about changing the mode using the three items above. This adaptability, combined with the underlying Reed-Solomon forward error correction (FEC) coding, and, when necessary, retransmission of erroneous frames, is the key to CLOVER’s reputation as one of the two fastest and most reliable HF data communication protocols available to hams. (The other is the PacTOR II system, which we have not yet tested.)

For further details on how CLOVER works, see the HAL documentation supplied with the various versions and the descriptions published by Ray Petit and Bill Henry in the *RTTY Journal*, *QST*, *QEX* and elsewhere between 1990 and 1994.

The P38 has been said to come close to the throughput of its more powerful CLOVER relatives on the basis of the claim that only the very best (high signal-to-noise ratio and little or no multipath) HF channels allow use of the 8P2A and 16P4A signalling constellations. To the extent that this claim is valid for our channels (it might apply on occasion to our NVIS paths), our results may describe CLOVER throughput performance in general. On very good paths, however, you might expect to see average throughput up to 25% higher than what we have measured if you use the PC14000 or DSP4100.

The NVIS paths we have studied, which are thirty to sixty miles long, frequently display strong multipath, high local and propagated noise, D-layer absorption at mid-day and occasionally strong interference from other stations operating in both voice and digital modes. (Horizontal antenna polarization at all our NVIS stations allowed us to be fairly certain we were using NVIS rather than surfacewave propagation, and this was confirmed by the fading we observed most of the time.)

We measured one-hop skywave throughput on a 600-mile path. This path produced less multipath than our three NVIS links and therefore somewhat higher throughput (see below).

The majority of NVIS measurements were at 3.6155 MHz LSB, with some at 1.8 15 MHz LSB. The OHS measurements were done at 10.141 MHz LSB. Output power (about 100 watts) and antennas were typical of those used by hams. Measurements were made over all daylight hours and a few were made in the evening while the maximum usable frequency (MUF) was still above the operating frequency. The NVIS tests covered the twelve-month period from May 1996 to May 1997. The OHS tests covered the eight-month period from October 1996 to June 1997. The average sunspot number during these periods was about ten, so MUFs were near their cyclical minima.

The rest of the paper describes the paths between stations and layout of antennas, the HAL P38 interface and file-transfer operation, the file types sent, the recorded data format, the statistical analysis software, a statistical summary of the data, a discussion of the statistical results and concluding remarks.

2. Layout of Paths and Discussion of Antennas

The stations used for the NVIS tests are in Bedford, Mass. (KB 1 JY), Norfolk, Mass. (WI IMM) and Derry, N.H. (KB1PZ). Bedford used an 80m dipole up 30 feet for most tests, and occasionally a terminated, bottom-fed 125-ft longwire pointing southwest (for NVIS tests on 160m). Norfolk used an 80m dipole up 40 feet or a longwire for 160m. Derry used an off-center-fed 80m dipole up 30 feet. The stations in the OHS tests are in Bedford, Mass. (KB 1 JY) and Raleigh, N.C. (KF4KL). In the OHS tests Bedford used the resistively terminated sloping

longwire running southwest. Raleigh used a 135-foot dipole running NW-SE and fed with ladder-line. The links (followed by lengths and rough estimates of the percentage of data collected over each link) are

NVIS

Bedford-Norfolk (35 miles, 45%)
Bedford-Derry (25 miles, 45%)
Derry-Norfolk (60 miles, 10%)

OHS

Bedford-Raleigh (600 miles, 100%)

(All of these links run more or less north-south.)

3. The HAL P38 User Interface and CLOVER File transfer

Although there are graphical user interfaces (GUIs) for running CLOVER that may be more sophisticated than the DOS-based one provided by HAL (Express and XPWIN, for example), the HAL GUI is presently the best one to use for making throughput measurements because it gives the transfer time of compressed file transfers. During the course of our tests we moved through versions 20 to 23 of the P38 firmware. (Some of the firmware upgrades made improvements in CLOVER's ARQ scheme.) The majority of our tests used Versions 21 and 22. The improvements appear to have been in the nature of fine-tuning since we have not noticed large increases in performance.

*

To send a file with the HAL GUI, one first establishes a CLOVER ARQ link with the receiving station by selecting (or entering) the callsign of the station and sending a carriage return.

Once the link is established (which is usually confirmed in a few seconds by a `LINKED` message on the GUI and the appearance of "HIS" channel statistics on the display), one can enter the name of a file to be transferred. (The file needs to be in the P38 directory.) Since distribution of Revision 22 of the P38 firmware, channel statistics can be recorded during file transfers; these often provide fascinating insight into the workings of a transfer (see the references at the end for our paper on CLOVER channel statistics).

After the filename has been entered, one has a choice of sending a text file from the TX Buffer in uncompressed mode or sending any file from the P38 directory ("Send from Disk") in compressed mode. Only generic text can be sent in uncompressed mode; files with control characters in them will abort a transfer attempt. Files sent in compressed mode can have any format, although some files may not benefit from compression, and may even be expanded slightly by the compression process (more on this below).

Some people view the throughput of uncompressed files as the inherent, or "true" performance of the CLOVER (or any other) waveform, error-control scheme and file-transfer system. Others (such as ourselves) consider compression to be part of a protocol if it is provided as a constituent of the "common" interface provided with a modem's hardware. To satisfy both camps, we have measured throughput of both compressed and uncompressed files.

This opened a small box of bother, because it turned out that how much a file gets compressed by the HAL compression software (a dictionary-based algorithm contained in the PKWARE Library) depends on what kind of a file it is. "Data" files (defined as text files with regular structure) are compressed more than arbitrary text files, and certain graphics files with large monochromatic parts can be compressed to much less than 50% of their original size. (See below.)

To measure the throughput of uncompressed (text) files, we had to record transfer time in our transfer-data files by hand, since the HAL software does not display transfer times of data sent from the text buffer. (This was not as onerous as it sounds since once we decided on a recording format we were able copy, paste and edit with a text editor to speed up data recording. We could, as an alternative, have written a program using HAL's control-code specification and developer guidelines to automate throughput testing, but we didn't have the leisure for that.)

To measure the throughput of compressed (text or other) files, we took advantage of the HAL software's calculation and display of file transfer time on the GUI screen. Through experimentation we deduced that the optimal (highest throughput with smallest test time) file size for throughput assessment of CLOVER was between 10 and 30k bytes, and most of our files had sizes in that range.

Precompressed files (for example, GZIP-ed ones) may be be sent by CLOVER in even smaller form than those sent compressed only by CLOVER's PKWARE algorithm, but this does not correspond to use of the "common" implementation. Nevertheless, precompression in HF data communication is a very useful technique that deserves further study.

4. File Types Sent

We have sent uncompressed and compressed text files, uncompressed and compressed "data" files, graphics files and "hybrid" files. Data files are defined to be fairly rigidly formatted text files that have a relatively large number of repeated strings and relatively short lines. An example of such a file is a CLOVER channel statistics file (recorded on command by the HAL interface). A snippet of a channel stats file is given below.

```
034601,KB1JY    ,BPSM   ,2,02,000, 000,000,000,000
034601,W1IMM    ,BPSM   ,2,02,031, 003,040,050,000
034604,KB1JY    ,BPSM   ,2,02,000, 000,000,000,000
034604,W1IMM    ,BPSM   ,2,02,028, 005,045,100,000
034606,KB1JY    ,BPSM   ,2,02,036,-6   ,054,000,000
034606,W1IMM    ,BPSM   ,2,02,026, 003,042,050,000
034609,KB1JY    ,BPSM   ,2,02,033,-3   ,049,000,000
034609,W1IMM    ,BPSM   ,2,02,026, 002,044,000,000
```

Graphics files come in many formats (*.bmp, *.gif, pict, etc.). Some formats (JPEG, MOV) usually come already compressed and get bigger when sent by CLOVER. Other graphics

formats (*.bmp, for example) can be compressed by CLOVER's PKWARE library and sometimes get much smaller (with resulting very high throughput) if they have large one-color sections with no detail in them. Generally speaking, the more detailed in layout and color a graphics file is, the less it can be compressed.

"Hybrid" files are those we define as having combinations of text and unprintable control characters. Examples of such files are Microsoft Word and Microsoft Excel files. Because the effect of compression on a hybrid file depends on both its layout and character content, the percent of compression of a hybrid file can be hard to predict. In our discussion of statistical results we will comment on the effect of file type on compression ratio and throughput.

Some files, like zipped files and executables, apparently look to most compression schemes like random sequences of characters and cannot be compressed much by the algorithms of PKWARE. These files may even be expanded somewhat by addition of useless compression-implementing strings. We did not include such files among those we sent. Their throughput would probably be about the same as that of uncompressed text files of similar size (see below).

5. Recorded Throughput-Data Format

The data archive file into which the results of each transfer are entered contains the file type ("g" for graphics, "d" for data, "h" for hybrid; see above), date-time group, callsign of receiving station, callsign of sender, CLOVER interface (P38 = HAL), uncompressed file size in bytes, compressed file size in bytes, transfer time in seconds, predominant waveform used by CLOVER during the transfer, HF frequency, observed channel condition (Q = quiet, etc.) hand-calculated throughput in characters per second (cps, which doesn't have to be accurate since it's calculated later by the analysis software) and the CLOVER "bias" chosen by the sender (F[ast], N[ormal] or R[obust]). In the case of text (including "data") files sent from the TX buffer, the transfer time is read from the computer clock and may therefore be off by a second or two. This makes little difference to throughput calculations for files such as ours that take several minutes to send.

The bias is a parameter that describes the error-correcting overhead (and thus the number of data-bytes per ARQ block) chosen by the transmitting station's operator for a communications session. Robust bias is usually chosen when the channel appears (sounds) bad and fast bias when it appears to be good.

Here's an excerpt from the NVIS transfer-data file for tests run in May 1997:

```
h16.05.97 20:54:00 WlIMM KB1JY P38 37376 10063 332 8PSM 3.6155 Q [113_cps]F
h16.05.97 20:59:00 WlIMM KBlJY P38 18944 6476  229 8PSM 3.6155 Q [83_cps]F
h16.05.97 22:05:00 WlIMM KBlJY P38 37376 10063 333 8PSM 3.6155 Q [112_cps]F
h16.05.97 22:09:00 WlIMM KBlJY P38 18944 6476  246 8PSM 3.6155 Q [77_cps]F
h21.05.97 12:29:00 KBlPZ KBlJY P38 37376 10063 343 8PSM 3.6155 Q [109_cps]F
h21.05.97 12:33:00 KBlPZ KBlJY P38 18944 6476  215 8PSM 3.6155 Q [88_cps]F
h21.05.97 12:40:00 KBlPZ KBlJY P38 37376 10063 341 8PSM 3.6155 Q [110_cps]F
h21.05.97 12:44:00 KBlPZ KBlJY P38 18944 6476  217 8PSM 3.6155 Q [87_cps]F
d21.05.97 12:59:00 KBlPZ KBlJY P38 26148 26148 915 8PSM 3.6155 Q [29_cps]F
d21.05.97 13:04:00 KBlPZ KBlJY P38 26148 6502  228 8PSM 3.6155 Q [115_cps]F
```

This file is opened and analyzed by a data-analysis program described in the next section.

6. The Data-analysis Software

The results in the data archive are analyzed off-line by a program called summary.c. This program reads the archive file line-by-line looking for various strings. As it moves through the file to the end-of-file character, the program keeps running totals of throughput and other data corresponding to the strings, from which it calculates statistics such as the average and standard deviation of the throughput. The statistics are written to a summary file after the pass through the archive file. Switches in the summary code are set before each run to pick out specific data (corresponding to various string combinations) for analysis. For example, we select lines starting with the letter "d" to calculate the throughput for "data" files (see below). Since the summary program was written to analyze archive files of fixed format but arbitrary length, summaries of the data collected so far can be made at any time.

Shown below is the output of the summary program for all P38 NVIS tests run from May 1996 to May 1997. For this output we set the software switches to compute throughput statistics for *uncompressed text files*.

```
Statistical Summary of P38-CLOVER Throughput Tests:
21.06.97 21:34:00 UNCOMPRESSED TEXT

NUMBER OF P38 TRANSFERS IN SAMPLE = 193
E(FILE_SIZE) = 13917.9 bytes, E(COMPRESSED_SIZE) = 13917.9 bytes
E(TRANSFER TIME) =  652.3 s, sd(TRANSFER TIME) =  385.5 s
E(THRUPUT) = 24.40 cps,sd(THRUPUT) = 6.80 cps, sd(mean_THRUPUT) = 0.490 cps
MAXIMUM THRUPUT = 35.71 cps, E(THRUPUT/Hz) = 0.049 cps/Hz
E(COMPRESSION RATIO) = 100.00%, sd(COMPRESSION RATIO) = 0.00%
Lowest compression ratio = 100.00%; Compressed-size:File_size = 0:0
Highest compression ratio = 100.00%; Compressed-size:File_size = 10000:10000
NUMBER OF UNCOMPRESSED TRANSFERS IN SAMPLE = 193
```

The output shows that the average throughput for 193 uncompressed-text-file transfers was about 24 characters per second (cps) and that the largest observed throughput in this mode was about 36 cps. The sd(THRUPUT) reflects the spread of the throughput measurements about their average. Roughly speaking, about two-thirds of a set of measurements will be within one standard deviation (here 6.8 cps) of their mean and over 90% will be within two standard deviations of their mean.

We also calculate the "standard deviation of the mean throughput" [sd(mean THRUPUT)] in characters per second and the average throughput per Hertz of signaling bandwidth. The standard deviation of the mean throughput (equal to the standard deviation of the throughput divided by the square root of the sample size) is an assessment of the variability of the mean itself (which has its own statistical variability). The sd(mean) above suggests that our sample size in this case is big enough to give us pretty high confidence that if we collected many more throughput measurements under roughly the same conditions, we would not get an average throughput that differed from the one above by more than about half a character per second.

To estimate the average throughput per Hertz [E(THRUPUT/Hz)], we divide the average throughput by the signaling bandwidth. For P38 CLOVER, the signaling bandwidth is 500 Hz (see the Clover documentation).

The compression ratio is defined as the ratio of compressed size in bytes to uncompressed size in bytes. A ratio of 100% therefore means no compression.

Here's the corresponding statistical summary for *compressed text files* sent over NVIS links.

```
Statistical Summary of P38-CLOVER Throughput Tests:
11.08.97 08:26:44 COMPRESSED TEXT

NUMBER OF P38 TRANSFERS IN SAMPLE = 266
E(FILE_SIZE)  = 19434.6 bytes,  E(COMPRESSED_SIZE) = 9958.6 bytes
E(TRANSFER TIME) =  468.1 s,  sd(TRANSFER TIME) =  284.0 s
E(THRUPUT) = 43.13 cps, sd(THRUPUT) = 11.30 cps, sd(mean_THRUPUT) = 0.693 cps
MAXIMUM THRUPUT = 65.57 cps, E(THRUPUT/Hz) = 0.086 cps/Hz
E(COMPRESSION RATIO) = 51.41%, sd(COMPRESSION RATIO) = 2.16%
Lowest compression ratio = 46.76%;  Compressed_size:File_size = 18316:39168
Highest compression ratio = 69.90%;  Compressed_size:File_size = 699:1000
NUMBER OF UNCOMPRESSED TRANSFERS IN SAMPLE = 0
```

Note that the text files we sent in compressed mode were reduced on average to about half (5 1.4%) of their original size, a fact reflected in the average throughputs for compressed and uncompressed text files.

7. Statistical Summary of Throughput Results

The rest of the results of our NVIS and OHS tests (as of summer 1997) are summarized in Tables 1 and 2 below. The first column gives the average throughput and its standard deviation, the average throughput per Hertz, the standard deviation of the mean throughput and the maximum observed throughput. The second column gives the number of transfers in each case. The third column gives the mean and standard deviation of the compression ratio for compressed transfers. The fourth column gives the mean and standard deviation of the transfer time and the fifth column the average number of bytes in the original, uncompressed files.

Table 1. Statistical Summary of P38-CLOVER NVIS Throughput Data

File Type	E(thruput) sd(thruput) E(tput/Hz) sd_mn(tput) max_tput	No. Xfers	E(Compr. Ratio) (CR) sd(CR)	E(xfer_tm) sd(xfer_tm)	E(No_char)
Uncompr. Text	24.4 cps 6.8 cps 0.05 cps/Hz 0.49 cps 35.7 cps	193	—	652 s 386 s	13918
Compr. Text	43.1 cps 11.3 cps 0.09 cps/Hz 0.69 cps 65.6 cps	266	51.4% 2.2%	468 s 284 s	19435
Uncompr. Data	26.3 cps 6.1 cps 0.05 cps/Hz 0.53 cps 35.4 cps	130	—	704 s 329 s	17935
Compr. Data	89.8 cps 29.2 cps 0.18 cps/Hz 2.5 cps 149.9 cps	135	26.6% 6.8%	315 s 178 s	24526
Graphics	75.3 cps 62.6 cps 0.15 cps/Hz 5.6 cps 392.3 cps	124	40.4% 14.4%	583 s 418 s	42100
Hybrid	88.9 cps 24.2 cps 0.18 cps/Hz 2.6 cps 234.3 cps	85	29.8% 4.4%	336 s 106 s	29130

Table 2. Statistical Summary of P38-CLOVER OHS Throughput Data

File Type	E(thruput) sd(thruput) E(tput/Hz) sd_mn(tput) max_tput	No. Xfers	E(Compr. Ratio) (CR) sd(CR)	E(xfer_ tm) sd(xfer_tm)	E(No_char)
Uncompr. Text	29.4 cps 4.0 cps 0.06 cps/Hz 0.57 cps 35.1 cps	49	—	542 s 206 s	15916
Compr. Text	50.4 cps 9.9 cps 0.10 cps/Hz 1.32 cps 62.0 cps	57	51.8% 3.1%	309 s 177s	16053
Uncompr. Data	30.0 cps 4.9 cps 0.06 cps/Hz 0.58 cps 50.6 cps	73	—	746 s 258 s	21640
Compr. Data	101.8 cps 19.8 cps 0.20 cps/Hz 2.3 cps 133.6 cps	71	25.1% 5.7%	236 s 85 s	22929
Graphics	100.6 cps 95.0 cps 0.20 cps/Hz 11.3 cps 396.4 cps	71	39.8% 14.5%	463 s 248 s	49255
Hybrid	93.5 cps 21.8 cps 0.19 cps/Hz 2.8 cps 197.5 cps	62	29.9% 4.8%	302 s 66 s	28449

8. Discussion of Results

Tables 1 and 2 show that the inherent (no compression used) over-the-air average throughput of P38 CLOVER on our links is around 24 cps on NVIS paths and 29 cps on the OHS path for daytime operations. (The standard deviations of these mean throughputs are about 0.5 and 0.6

cps, giving us high confidence that additional measurements made under the same conditions would yield nearly the same mean throughputs.)

At night, especially with the sunspot cycle near minimum, we have found that on our paths there is usually so much multipath, noise and interference on the small band of frequencies that lie below the MUF in the ham bands that throughput is generally significantly lower than the daytime values (most of the ones presented here). (An automatic link establishment [ALE] system, such as prescribed in MIL-STD-188-141A, and now widely available, can often find a useful, interference-free frequency even at night if given enough choices and a suitable-usually broadband-antenna.)

The average P38 throughputs for compressed text files are 43 and 50 cps on our NVIS and OHS paths, with standard deviations of mean throughput of about 0.7 and 1.3 cps. These average throughputs, and the average compression ratios of about 52% given in column 4 of the tables, show that CLOVER's PKWARE compression is squeezing our text files down to about half their uncompressed size. This, of course, doubles text-file throughput.

In contrast, PacTOR (with Huffman compression on) and GTOR (with its built-in compression) achieve average throughputs for text files of about 18 and 24 cps on NVIS links and 20 and 32 cps over OHS links (see Ref. 4). Thus, if data compression is viewed as an intrinsic part of what we have called (Ref. 4) the "common" implementation of PacTOR and GTOR, then P38 CLOVER in its common (PKWARE compression on) implementation has about twice the average throughput of PacTOR and GTOR for text-file transfers. Average P38-CLOVER throughput *without* compression is about the same as GTOR (with built-in compression) and a bit higher than PacTOR (with its Huffman compression turned on). P38-CLOVER throughput without compression is almost double that of PacTOR with Huffman off.

The throughput of uncompressed "data" files is about the same as that of ordinary text files, since these are indistinguishable to CLOVER's text-buffer transfer mode. However, because of their characteristic repetitions of strings such as placeholders for null-entries (see the CLOVER channel statistics excerpt given above as an example of a data file), and their generally short lines, data files usually get compressed by more than 50% (i.e., their compression ratios by our definition are less than 50%). The data files we have sent got compressed to about a quarter of their original size with resulting average throughputs of 90 and 102 cps over NVIS and OHS paths.

Our graphics files ranged in size from 9k to 230k bytes. (CLOVER forces one to send graphics files in compressed mode ("Send-from-Disk") since the uncompressed send-from-buffer mode can only handle text files. Some of these graphics files were not very complicated; others were relatively detailed. The average compression ratio for graphics files was about 40% and the average NVIS and OHS throughputs were 75 and 101 cps.

Recall that we define "hybrid" files as those with a combination of text and control (e.g., formatting) characters. Examples are word-processor and spreadsheet files. Our hybrid files ranged in size from 1 Ok to 37k bytes. (CLOVER also forces one to send such hybrid files in

compressed mode.) Because of their often complicated layout and unpredictable distribution of repeated characters, it's hard to guess compression ratios accurately for hybrid files. The average compression ratio of the ones we sent (Microsoft Word and Excel files) was about 30%. The NVIS and OHS throughputs of our hybrid files were 89 and 94 cps over NVIS and OHS paths.

The relatively large standard deviations of uncompressed and compressed file throughput for all file types (the standard deviations range from four to several tens of cps) reflect, in part, CLOVER's ability to adapt itself to changing channel conditions (see Sec. 1). However, these standard deviations are also affected by variability of file size (especially for graphics files), so channel adaptation should not be viewed as the sole source of throughput spread.

9. Concluding Remarks

We hope that our data will aid understanding of CLOVER file transfer over HF and perhaps serve as a useful introduction to how CLOVER works. CLOVER is now used all over the world by amateurs (in particular, for BBS mail forwarding). A number of international aid and other communications-providing organizations also use the P38 and other CLOVER versions (some with two-kHz bandwidth) to send medical and other information across parts of Africa and Australia, where alternative means of long-haul communication are not available, cr too expensive. CLOVER modems are also used for at least two special-purpose e-mail systems (one used widely by private boating and shipping operators). Our data may shed light on why this inexpensive, amateur-developed modem and its protocols have become so popular.

Acknowledgments

We are grateful to Doug Hall (KF4KL) for his customary reliability in keeping a P38 on the air in Raleigh, and to Bill Henry (K9GWT) and Drew White (K9CW) of HAL for useful comments on the paper. (The work was not supported financially by HAL.)

References

1. Ken Wickwire (KB1 JY), "A Software Package for Analyzing CLOVER Channel Statistics." To appear in *QEX,* October 1997.

2. Ken Wickwire (KB1 JY) et al., "On-air Measurements of HF TOR and Packet Throughput, Part I: Near-Vertical-Incidence-Skywave Paths," *Digital Journal,* March 1996.

3. Ken Wickwire (KB 1 JY), "On-air Measurements of HF TOR and Packet Throughput, Part II: One-hop Skywave Paths," *QEX,* June 1996.

4. Ken Wickwire (KB1 JY), "On-air Measurements of HF Data Throughput: Results and Reflections," *Proc. 15th ARRL/TAPR Digital Communications Conference,* ARRL, 1996.

Notes

Notes